近线性时间求解带负权单源最短路径：更快了！

Karl Bringmann萨尔兰大学与马克斯·普朗克信息学研究所，萨尔兰信息学园区，德国萨尔布吕肯

Alejandro Cassis萨尔兰大学与马克斯·普朗克信息学研究所，萨尔兰信息学园区，德国萨尔布吕肯

Nick Fischer
魏茨曼科学研究所，以色列雷霍沃特

arXiv:2304.05279v1 [cs.DS] 11 Apr 2023

摘要

在这项工作中，我们重新研究了基础的单源最短路径（SSSP）问题，其中边权可能为负。Bernstein、Nanongkai 和 Wulff-Nilsen 最近取得突破性成果，针对负权单源最短路径问题提出了一种近线性的 $O(m \log^8(n) \log(W))$ 时间算法，其中 W 是最小负权边绝对值的上界。在这项工作中，我们将运行时间改进至 $O(m \log^2(n) \log(nW) \log\log n)$，提升幅度接近六个对数因子。其中一些对数因子很容易消去（例如，替换 Dijkstra 算法中使用的优先队列），而另一些则要复杂得多（例如，为了寻找负环，我们设计了一种类似带噪二分搜索的算法，并通过漂移分析对其进行分析）。

作为附带结果，我们得到了一种以相同运行时间以及用于计算有向图中低直径分解的新构造

2012 ACM 学科分类计算理论 → 图算法

关键词和短语最短路径、低直径分解、漂移分析

资助 本工作是项目 TIPEA的一部分。该项目获得了欧洲研究委员会（ERC）在欧洲联盟“地平线2020”研究与创新计划下的资助（资助协议编号 850979）。

尼克·菲舍尔：这项工作的部分内容是在作者任职于萨尔兰大学和马克斯·普朗克信息学研究所期间完成的。本工作是 CONJEXITY项目的一部分，该项目获得了欧洲研究委员会（ERC）在欧盟“地平线欧洲”研究与创新计划下的资助（资助协议编号：101078482）。

致谢 感谢 Danupon Nanongkai 就这一主题进行的许多有益讨论。

1 引言

几个世纪以来，计算最短路径的探索一直吸引着数学家和计算机科学家的关注。在这项工作中，我们重新研究带有可能为负的边权的单源最短路径（SSSP）问题：给定一个有向加权图 G 和一个指定的源顶点 s，计算从 s 到 G中所有其他顶点的距离。这可能是最基础的加权图问题之一，在计算机科学中有着广泛的应用，包括路由、数据网络、人工智能、规划和运筹学。

众所周知，近 60 年来，对于边权非负的单源最短路径问题，Dijkstra 算法可以在近线性时间内求解 [23, 62, 29], ；而负权情形的发展历程则更加引人入胜：贝尔曼-福特算法诞生于 50 年代 [54, 26, 9, 46] ，运行时间为 $O(mn)$，这一时间界长期以来一直是当时的最佳结果。从 80 年代开始，人们发展出了缩放技术，并

将运行时间降至 $O(\overline{m}\sqrt{n}\log W)$ [32, 33, 34]；这里以及下文中，W表示图中最小负权边权重的绝对值。[1] 其他论文则聚焦于特定类型的图，例如为平面有向图提出了近线性时间算法 [43, 38, 24, 37]，并改进了小权重稠密图的算法 [53]。

另一种方法是将单源最短路径建模为最小费用流问题。[2] 在过去十年中，凸优化技术与动态算法的结合推动了最小费用流计算的一系列进展 [21, 7, 59, 58]，从而也推动了负权单源最短路径方面的发展，其运行时间为 $\widetilde{O}(m^{10/7})$ [21]、$\widetilde{O}(m^{4/3})$ [7]以及 $\widetilde{O}(m+n^{3/2})$ [59]。这一研究方向最近由 Chen、Kyng、Liu、Peng、ProbstGutenberg 和 Sachdeva 推动，最终得到了一个近乎线性 $m^{1+o(1)}$-时间算法 [17]。

最后，在最小费用流计算取得突破的同时，Bernstein、Nanongkai 和 Wulff-Nilsen [11] 找到了一个令人惊叹的近线性 $O(m\log^8(n)\log(W))$ 时间负权单源最短路径算法。我们将他们的算法称为*BNW* 算法。BNW 算法是组合式的，而且可以说相当简单，因此令人满意地回答了负权单源最短路径问题的粗粒度复杂性问题。然而，故事并未就此结束。在这项工作中，我们进一步深入探究 Bernstein 等人 [11]留下的以下开放问题：

> 我们能否进一步减少负权单源最短路径运行时间中的对数因子？

作为比较，非负权单源最短路径问题在上个世纪经历了一系列低阶改进 [23, 29, 30, 31, 56, 51, 52, 60, 61, 44, 2, 20, 57]，其中包括对数因子、甚至仅对数对数因子的改进。[3] 本着同样的精神，我们开始对负权最短路径中的低阶因子进行精细化研究。

1.1 我们的结果

在主要结果中，我们针对核心问题取得了显著进展，并将 BNW 算法的运行时间缩短了近六个对数因子：

▶**定理 1** (负权单源最短路径)。 存在一种拉斯维加斯算法，给定一个有向图 G 和一个源点 s，要么计算出从 s 出发的最短路径树，要么在 G中找到一个负环，其运行时间为 $O((m+n$ 对数对数 $n)$ 对数^2 n 对数$(nW))$，且以高概率（并且在期望意义下）达到此运行时间。

我们通过优化 BNW 算法，将其推向极限来获得这一结果。Aaron Bernstein 在介绍他们的研究成果时指出，“类似于 log^5 的因素是其[当前框架中固有的] ”。[4] 因此，令人惊讶的是，我们在同一更广泛的框架内，将性能大幅提升至接近三个对数因子。尽管速度有所提升，我们的

[1] 严格来说，$W \geq 0$ 是使所有边权满足 $w(e) \geq -W$ 的最小数。略微滥用记号时，我们通常写作 $O(\log W)$ 来表示 $O(\max\{1, \log W\})$。[2] 要将单源最短路径建模为最小费用流问题，将每条权重为 $w(e)$ 的边 e 解释为一条容量无限、费用为 $w(e)$ 的边。此外，在图中添加一个人工汇点 t，并从所有顶点 v 向 t添加容量为 1、费用为 0 的边。这样，任何将 n 个单位从 s路由到 t 的最小费用流，都恰好对应于原始图中的一棵最短路径树（假设其中不包含负权环）。[3] 在这些论文中，Dijkstra 算法的运行时间 $O(m+n\log n)$ 被改进到了当前的最佳结果 $O(m+n\log\log\min\{n, C\})$ [57],，其中 C是图中的最大权重。

[4] https://youtu.be/Bpw3yqWT_d0?t=3721

算法的核心仍然是模块化且简单的。在第1.2节中，我们详细讨论我们的算法与 BNW 算法之间的技术相似之处和差异。

回想一下，只有在不存在负环的图中，计算最短路径才有意义（否则，两个节点之间可能存在一条权重任意小的负值路径）。鉴于此，我们在定理 1 中以尽可能强的形式解决负权单源最短路径问题：该算法要么返回一棵最短路径树，要么返回一个负环，作为不存在最短路径树的证明。事实上，检测负环这一子问题本身在文献中已受到广泛关注（参见综述 [19]）。

存在负环时，除了寻找一个这样的环，还有一种自然的替代方案：无论如何都计算图中的所有距离（其中一些距离为 $-\infty$ 或 ∞）。这一任务可以在相同的运行时间内解决：

▶ **定理 2** (负权单源距离)。存在一个拉斯维加斯算法，给定有向图 G 和源点 $s \in V(G)$，可计算从 s 到图中所有其他顶点的距离（这些距离可能为 $-\infty$ 或 ∞），运行时间为$O((m + n \log\log n) \log^2 n \log(nW))$，且以高概率（以及在期望意义下）成立。

鉴于最短路径问题有无数实际应用，一个值得探讨的重要问题是：是否存在一种具有竞争力的实现，能够处理负权重最短路径问题？竞赛编程中常见的实际实现会采用贝尔曼-福特算法的优化变体，例如“最短路径加速算法” [46, 22]（另请参阅 [18]，了解对贝尔曼-福特其他更复杂变体的实验评估）。然而，很容易找到一些实例，使得这些算法需要 $\Omega(mn)$ 的时间。经过数十年的竞赛编程，如果终于能有一种高效实现来处理这些实例，那将令人振奋。BNW 算法有九个对数因子，因此并不适合作为实用候选方案。我们相信，我们的工作为实现一种速度相当快的算法铺平了道路。

除了主要结果之外，我们还在两个密切相关的问题上取得了进展：计算最小环均值，以及有向图中的低直径分解。我们将在接下来的章节中介绍这些结果。

1.1.1 最小环均值

在一个有向加权图中，均值环 C 的定义为比值 $\bar{w}(C) = w(C)/|C|$其中 $w(C)$是 C 的总权重。最小环均值问题是：给定一个有向加权图，计算所有环中的最小均值，即 $\min_C \bar{w}(C)$。这是网络流 [1],领域中的一个核心问题，应用于验证和反应式系统分析 [15]。

关于计算最小环均值的研究文献非常丰富。1987年，Karp [36] 提出了一个 $O(mn)$时间算法，至今仍是运行时间最快的强多项式时间算法。对于弱多项式算法，Lawler 指出，该问题等价于检测负环，运行时间相差一个因子 $O(\log(nW))$ [40, 39]。事实上，其中一个方向显而易见：当且仅当最小环均值为负时，图中才存在负环。对于另一个方向，他将该问题归约为在 $O(\log(nW))$个边权经过修改的有理数图中检测负环。因此，根据 Lawler 的观察，可以通过黑盒方式将任何负权单源最短路径算法转化为最小环均值算法，运行时间开销为 $O(\log(nW))$。

此外，针对最小环均值计算也有专门的结果：Orlin 和 Ahuja [47] 设计了一种运行时间为 $O(m\sqrt{n}\log(nW))$ 的算法（优于基准

$O(m\sqrt{n}\log^2(nW))$，该基准由 [32, 33, 34] 的单源最短路径算法得出）。对于权重为 0-1 的特殊稠密图，已知一种运行时间为 $O(n^2)$ 的算法 [14]。最后，就近似算法而言，已知可以在$\tilde{}$时间内计算一个 $(1+\varepsilon)$-近似，运行时间为 $O(n^\omega \log(W)/\varepsilon)$ [15]。

对于带负权的单源最短路径问题，所有这些算法都不如最新的 BNW 算法：根据 Lawler 的观察，他们的算法能在 $O(m\log^8(n)\log^2(nW))$ 时间内计算最小环均值。事实上，他们的工作隐含地表明，运行时间可以降至 $O(m\log^8(n)\log(nW))$。我们的贡献同样是将对数因子的数量从九个减少到近三个：

▶**定理 3** (最小环均值)。 存在一种拉斯维加斯算法，给定一个有向图 G ，可以找到平均权重最小的环 C $\bar{w}(C) = \text{最小}_{C'}\,\bar{w}(C')$，运行时间为 $O((m+n\log\log n)\log^2 n\log(nW))$，且该算法以高概率（并且在期望意义下）完成。

1.1.2 有向低直径分解

BNW 算法的一个关键组成部分是有向图中的低直径分解（LDD）。我们的单源最短路径算法在这一点上不同于 BNW 算法，并未显式使用低直径分解。然而，作为本工作的一个附带成果，我们改进了有向图中已知的最佳低直径分解。

Awerbuch 近 40 年前首次研究了低直径分解 [3] ，此后低直径分解发现了多种应用，主要用于无向图，也主要应用于分布式、并行和动态场景
[5, 6, 4, 42, 8, 12, 45, 48, 27, 16, 10, 28, 11]。这些工作中的精确定义大多有所不同，但共同思想是选择一个较小的边子集 S，从图中移除 S 中的所有边后，剩余图具有直径有界的（强）连通分量。

对于有向图，我们区分两类低直径分解：弱 低直径分解确保，对于图 G中的每个强连通分量 $C \setminus S$， C 在原始图中的直径有界。强 低直径分解则具有更强的性质： C 在图 $G \setminus S$ 中的直径有界。

▶ **定义 4** （有向低直径分解）。 具有 ρ 开销的弱低直径分解是一种拉斯维加斯算法，给定一个具有非负边权 w 的有向图 G 和参数 $D>0$，计算边集 $S\subseteq E(G)$ ，并满足以下性质：

- 稀疏命中： 对于任意边 $e\in E$，$\mathbf{P}(e\in S)\le O(\frac{w(e)}{D}\cdot\rho+\frac{1}{\text{poly}(n)})$。
- 弱直径： 每个 *SCC* C 属于 $G\setminus S$，其弱直径至多为 D（也就是说，对于任意两个顶点 u、$v\in C$，都有 $\mathrm{dist}_G(u,v)\le D$） 。

我们称低直径分解是强的，如果它还满足以下更强的性质：

- 强直径： $G\setminus S$ 中的每个强连通分量 C 的直径至多为 D（也就是说，对于任意两个顶点 u、$v\in C$，我们有 $\mathrm{dist}^{G\setminus S}(u,v)\le D$） 。

对于有向图，最先进的弱 低直径分解由 Bernstein、Nanongkai和 Wulff-Nilsen [11] 开发，作为其最短路径算法的工具。他们的结果是一种具有多对数开销 $O(\log^2 n)$ 的弱低直径分解，其运行时间为近线性时间 $O(m\log^2 n+n\log^2 n\log\log n)$。就强低直径分解而言，目前尚无可比的结果。虽然将他们的算法改造为计算强低直径分解并不困难，但这种增强会导致更慢的运行时间$\Omega(nm)$。我们的贡献是设计出首个可在近线性时间内计算的强低直径分解，其开销仅略有增加 $O(\log^3 n)$：

▶ **定理 5** (强低直径分解)。 存在一个强低直径分解，其开销为 $O(\log^3 n)$，可在 $O((m+n\log\log n)\log^2 n)$ 时间内以高概率（以及在期望意义下）计算得到。

1.2 技术概述

我们的算法受BNW 算法的启发，并遵循其总体框架，但在许多方面有所不同。本节将对此进行详细比较。

1.2.1 框架

我们对算法的介绍采用模块化方式：首先，我们将重点讨论受限类别图上的单源最短路径问题（下文简称为受限图，见下文定义 6）。第二步，我们将展示如何以黑盒方式归约到受限问题，从而得到一般图上的单源最短路径问题的结果、查找负环，以及计算最小环均值。

▶ **定义 6** (受限图)。带边权的有向图 $G=(V,E,w)$，其中指定的源顶点为 $s\in V$，若满足以下条件，则称其为受限：

- 边权为整数且至少为 -1。最小环均值至少为 1。源点 s 通过一条权
- 重为 0 的边与每个其他顶点相连。

特别要注意，受限图不包含负环，因此总是可以计算出一棵最短路径树。受限单源最短路径 问题是在给定的受限图 G 中计算一棵最短路径树。我们用 $T_{\mathrm{RSSSP}}(m,\ n)$ 表示错误概率为 $1\ 2$（比如说）的受限单源最短路径算法的最优运行时间。

1.2.2 改进 1：通过更好的分解加速受限单源最短路径

Bernstein 等人 [11] 证明了 $T_{\mathrm{RSSSP}}(m,\ n)\ =\ O(m\log^5 n)$。我们的首项贡献是将对数因子减少了近三个，并将这一界改进为 $T_{\mathrm{RSSSP}}(m,\ n)\ =O((m+n\log\log\ n)\log^2 n)$（见定理 18）。

从整体上看，BNW 算法的思路是通过寻找适合以下两个子任务的边子集 S，将图分解：(1) 我们可以在移除边集 S 后得到的图 $G\setminus S$ 中递归计算最短路径，从而取得足够进展，使总运行时间只产生较小的多对数开销。(2) 给定递归调用的结果，我们可以高效地将边 S “加回”，从而得到 G 的正确最短路径树。对于后一个任务，关键性质是 S 与 G 中的每条最短路径相交的次数至多为 $O(\log n)$（按期望计）；这样，Dijkstra 算法和贝尔曼-福特算法的一个简单推广就能在近线性时间内调整最短路径树（见引理 25）。

对于我们的结果，我们基本保留步骤 (2) 的实现不变，但由于 Thorup [57]（见引理 25）提出了一种更快的Dijkstra 算法实现，因此我们采用了该实现。最显著的差异出现在步骤 (1) 中，在这一步我们改变了算法选择 S 的方式。具体而言，Bernstein等人使用有向低直径分解来实现该分解。我们遵循相同的思路，但推导出了一种更高效、更直接的分解方案。为此，我们定义以下关键参数：

▶**定义 7**。 设 G 为一个指定源点 s的受限图。我们将 $\kappa(G)$ 定义为任意从 s 出发且权重非正 $w(P) \leq 0$的简单路径 P 中负权边（即权重恰好为 -1的边）的最大数量。

我们新的分解可以表述如下。

▶ **引理 8** (分解)。 设 G 是一个源顶点为 $s \in V(G)$ 且满足$\kappa \geq \kappa(G)$的受限图。存在一个随机化算法 *Decompose*(G, κ) ，其期望运行时间为$O((m + n\log\log n)\log n)$，可计算出一个边集 $S \subseteq E(G)$，满足：

1. 进展：以高概率，对于 $G\setminus S$ 中的任意强连通分量 C，我们有 *(i)* $|C| \leq \frac{3}{4}|V(G)|$ 或 *(ii)* $\kappa(G[C \cup \{s\}]) \leq \frac{\kappa}{2}$。*2.* 稀疏命中：对于 G 中任意最短的 s-v-路径 P，有 $E(|P \cap S|) \leq O(\log n)$。

稀疏命中性质正是我们处理 (2) 所需要的。借助进展条件，我们确保在 $G\setminus S$ 的强连通分量上递归时，$|V(G)| \cdot \kappa(G)$ 会按常数因子缩小。因此，递归树的深度至多为 $O(\log(n \cdot \kappa(G))) = O(\log n)$。总之，借助这一新思路，我们可以在 $O((m + n\log\log n)\log^2 n)$ 时间内求出受限图中的最短路径。

1.2.3 改进 2：更快的缩放

接下来，我们只需付出至多再增加一个对数因子 $\log(nW)$ 的代价，就能将受限单源最短路径算法提升到一般的单源最短路径问题。相比之下，BNW 算法在这里需要付出四个对数因子 $O(\log^3 n \log W)$。作为预热，我们假设给定的图保证不包含负环。

预热：从受限图到无负环的图 这项任务是一个非常适合采用80年代缩放技术的典型例子 [32, 33, 34]：通过将给定图 G中从 $w(e)$ 到 $\lceil 3w_{(e)}W+1 \rceil + 1$的权重进行舍入，我们确保(i) 所有权重都至少为 -1 ，并且(ii) 最小环均值至少为1，从而将 G 转换为一个受限图 H（见引理30）。我们计算 H 中的最短路径，并使用计算出的距离（借助势函数）来增加原始图 G中的权重。如果 G 的最小权重为 $-W$，通过这种方式，我们可以得到一个等价的图 G'，其最小权重为 $-\frac{3}{4}W$，其中等价性定义如下：

▶ **定义 9** (等价图)。 我们称同一组顶点和边上的两个图 G, G' 为等价，若 *(1)* G 中的任意最短路径也是 G'中的最短路径，反之亦然，且 *(2)* 对于任意环 C, $w_G(C) = w_{G'}(C)$。

因此，根据(1)，我们继续计算 G' 中的最短路径。乍一看，重复这一缩放步骤似乎只会使运行时间增加一个 $\log W$ 因子，但由于一些微妙的原因，实际开销是 $\log(nW)$。另一个问题是，受限单源最短路径算法以恒定概率出错。简单的修复方法由于增强而额外损失一个 $\log n$ 因子（Bernstein 等人正是通过这种方式得到其算法 SPMonteCarlo，参见 [11, 定理7.1]）。幸运的是，我们可以将缩放和增强步骤“合并”，从而将总开销降低到 $\log(nW)$，参见定理 29。

从受限图到任意图 如果 G 包含负环，该怎么办？此时，我们的目标是找出并返回一个这样的负环。除了显而易见的优点——让输出包含更多信息之外，这也使我们能够将算法从蒙特卡洛算法强化为拉斯维加斯算法，因为最短路径树和负环都可以作为

证书，且可以高效地进行检验。像之前一样使用缩放技术，我们可以轻松地检测给定的图是否包含负环，所需时间为 $O(T_{\mathrm{RSSSP}}(m,n)\cdot\log(nW))$（甚至可以以高概率完成，见推论 34），但我们无法找出这样的环。

我们给出了一种高效归约，可将寻找负环的问题归约到受限单源最短路径问题，开销为 $O(\log(nW))$。这一归约是本文中技术上最复杂的部分。接下来的段落将简要介绍其中的主要思路。

一种类似噪声二分搜索的问题 在本概述的其余部分中，我们会尽可能抽象地表述核心挑战，并省略更多背景信息。我们使用以下记号：给定一个有向图 G 和一个整数 M，我们用 G^{+M} 表示通过将 M 加到 G的每条边权上得到的图。考虑以下任务：

▶ **定义 10** (阈值)。给定一个带权图 G，求最小整数 $M^*\geq 0$，使得将 M^* 加到图 G 的所有边权上得到的图 G^{+M^*}不含负环。

我们的目标是在 $O(T_{\mathrm{RSSSP}}(m,n)\log(nW))$ 时间内解决阈值问题（由此可知，我们也能在相同时间内找到负环，见引理 36）。我们可以将以下引理作为黑箱工具使用（其证明与预热情形类似）：

▶ **引理11** (非正式引理 40)。存在一个 $O(T_{RSSSP}(m,n))$-时间算法，对于给定的图 G 及其最小权重 $-W$，该算法要么返回一个具有最小权重 $-\frac{3}{4}W$的等价图 G'，要么返回 *NegativeCycle*。如果 G不包含负环，则该算法返回 *NegativeCycle* 的错误概率至多为 0.01。

从本质上说，引理 11 提供了一种测试给定图 G 是否包含负环的方法。因此，一个自然的想法是使用引理11作为测试器，通过二分搜索找到 M^*。然而，请注意，这个测试器是单边的：如果 G 包含负环，那么测试器不一定会检测到它。幸运的是，我们可以将测试器转化为一种双赢算法来计算 M^*。

我们首先在一个理想化情形下介绍我们的阈值算法，在这种情形下，我们假设引理 11 中的测试器错误概率为 0。我们令 $d=\frac{1}{5}W$，并在图 G^{+d} 上运行测试器。存在两种情况：

- 测试器返回负环：在理想化情形下，我们可以假设 G^{+d}确实包含一个负环。因此，我们递归计算 G^{+d} 的阈值，并返回该值加上 d。注意，G^{+d} 的最小权重至少为 $-W+d=-\frac{4}{5}W$。测试器返回一个等价图 G'：在这种情况下，我们递归计算并返回 $(G')^{-d}$的阈值。注意，图 G 和 $(G')^{-d}$ 的阈值相同，因为根据定义 9，对于任意环 C，都有 $w_{(G')^{-d}}(C)=w_{G'}(C)-d=w_{G^{+d}}(C)-d=w_G(C)$。此外，由于 G^{+d}的权重最小，为 $-\frac{4}{5}W$，根据引理 11，等价图 G'的最小权重至少为 $-\frac{3}{4}\cdot\frac{4}{5}W=-\frac{3}{5}W$。因此，$(G')^{-d}$ 的最小权重至少为 $-\frac{3}{5}W-d=-\frac{4}{5}W$。

在这两种情况下，我们都会递归计算最小权重至少为 $-\frac{4}{5}W$ 的图的阈值。因此，递归会达到深度 $O(\log W)$，直到我们将图缩减为常数最小权重，此时问题就变得简单了。

上述算法在理想化情形下有效，但在非理想化情形下呢？此时测试器可能以常数概率出错。当然，我们可以先提升测试器的成功概率，使其以高概率成功。结合上述算法，这样就能在 $O(T_{\mathrm{RSSSP}}(m,n)\log(nW)\log n)$ 时间内解决阈值问题。

然而，这项任务真正的复杂性在于避免朴素的增强。通过精确理解具有常数错误概率的非理想化情形，我们将阈值的运行时间改进为 $O(T_{\mathrm{RSSSP}}(m,\ n)\log(nW))$。为此，似乎可以应用带噪二分搜索 的技术（例如参见 [49, 25, 50]）。遗憾的是，已知结果似乎不适用于我们的情形，因为引理 11仅提供了单侧测试器。我们对这一最终挑战的解决方案，是将上述概述的算法与带噪二分搜索中的思想进行创新性结合。分析利用了漂移分析（例如参见 [41]），其中包括定义一个合适的漂移函数（一种在每一步中期望减少一个常数因子的量，并且当且仅当我们找到最优值 M^* 时为零），以及应用漂移定理（参见定理 44）来证明漂移函数会快速趋近于零。

1.3 LogShaves 总结

最后，为了便于与BNW 算法进行比较，我们简要列出具体在哪些地方削减了近六个对数因子。我们先从受限单源最短路径算法的改进说起：

- 我们使用 Thorup 优先队列 [57] 来加速 Dijkstra 算法，见引理 25。这将一个对数因子的开销降为对数对数因子。与 BNW 算法中由低直径分解带来的 $O(\log^2 n)$ 开销相比，
- 我们的分解方案的稀疏命中性质（引理 8）仅带来$O(\log n)$ 的开销。我们的分解方案的进展性质（引理 8）确保受限单源最短路径算法的递归深度仅为 $O(\log n)$。BNW 算法中相应的递归深度为 $O(\log^2 n)$（减少节点数量所需的深度为 $\log n$，乘以减少 $\max_v \eta_G(v)$ 所需的深度 $\log n$）。

接下来，我们总结在缩放步骤中削减的对数因子：

- BNW 算法通过重复运行受限单源最短路径算法 $O(\log n)$ 次来提高其成功概率。我们将增强与缩放步骤结合起来，从而省去了这一对数因子。我们改进了从寻找负环到受限单源最短路径的整体归约。特别是，我们给出了阈值的一种实现，其速度快了两个对数因
- 子（见引理 36 和 42）。为此，我们通过漂移函数进行了复杂的分析。

1.4 未解决问题

我们的工作留下了几个有趣的开放问题。我们的负权单源最短路径算法还能进一步改进吗？具体来说：
1. $\log n$ 因子的数量还能进一步改进吗？在我们的算法中，我们受到三个log因子的影响：(i)缩放技术（$\log(nW)$），它将问题归约为受限图；以及在受限图上，(ii) 图分解固有的 $\log n$ 开销；(iii) 递归深度$\log n$，用于逐步归约 $\kappa(G)$。这些因素似乎都无法避免。因此，我们认为，如果不大幅改变框架，就很难改进我们的算法。 2. 缩放技术造成的损失能否从 $\log(nW)$降低到 $\log\ W$？经典的缩放技术作为一种归约，将问题转化为权重至少为 -1 的图时，只需要$\log W$ 次迭代 [34]。但在我们的设置中，由于受限图的条件更强（以及增强的作用），我们需要 $\log(nW)$ 次迭代。我们能做得更好吗？

3. 能否彻底去掉运行时间中的 $\log W$ 因子？也就是说，是否存在一个 强多项式 近线性时间算法？对于非缩放算法，贝尔曼-福特算法仍然是最先进的算法，运行时间为 $O(nm)$。这个问题已被反复提出，似乎非常难以解决。4. 能否将该算法去随机化？对于负权单源最短路径，目前最快的确定性算法仍然是 $O(m\sqrt{n}\log(W))$算法，由 [32, 33, 34] 提出；寻找近线性时间算法仍是一个开放问题。

1.5 概述

本文结构如下。第 2 节给出一些形式化的预备知识。第 3 节介绍我们用于受限图中负权单源最短路径的算法。第 4 节将上一节的算法扩展到不含负环的一般图。第 5 节去除这一假设，并增强算法，使其能够寻找负环，同时不增加运行时间。第 6 节给出我们关于计算最小环均值的结果。最后，第 7 节介绍我们在有向图低直径分解方面改进后的结果。

2 预备知识

我们记作 $[n]=\{1,\dots,n\}$和 $\widetilde{O}(T)=T\cdot(\log T)^{O(1)}$。若某事件以概率 $1-1/n^c$发生，且其中的常数 c可以任意大（此处， n表示输入图中的顶点数），则称该事件以高概率发生。除非另有说明，我们的算法均为蒙特卡洛算法，并以高概率成功。

有向图。本文中，我们始终考虑有向边权图 $G=(V,E,w)$。其中 $V=V(G)$是顶点集， $E=E(G)\subseteq V(G)^2$是边集。所有边权均为整数，记作 $w(e)=w(u,v)$，其中 $e=(u,v)\in E(G)$。通常设 $n=|V(G)|$且 $m=|E(G)|$。我们用 $G[C]$ 表示顶点为$C\subseteq V(G)$的诱导子图，并用 $G\setminus S$ 表示删除 $S\subseteq E$中所有边后得到的图 G。我们用 $\deg(v)$表示 v的（出）度，即从 v出发的边数。

A 强连通分量（*SCC*）是一个极大的顶点集合 $C\subseteq V$（G），其中任意一对顶点都可以相互到达。众所周知，每个有向图都可以 Decompose 为一组强连通分量，将强连通分量压缩为单个节点后得到的图是无环的。众所周知，还可以在线性时间内计算出强连通分量：

▶**引理 12**（强连通分量， [55]）。对于任意有向图 G，可以在 $O(n+m)$ 时间内识别其强连通分量。

对于边集 S （例如一条路径或一个环），我们记作 $w(S)=\sum_{e\in S}w(e)$。负环是满足 $w(C)<0$的环 C。对于顶点 u,v，我们用 $\mathrm{dist}_G(u,v)$表示最短 u-v路径的长度。如果某条 u-v路径中存在负权环，则令 $\mathrm{dist}_G(u,v)=-\infty$；如果不存在从 u到 v的路径，则令 $\mathrm{dist}_G(u,v)=\infty$。

▶定义 *13*（球）。对于一个顶点 v和一个非负整数 r，我们将以 v为中心、半径为 r的出球记为 $B^{out}_G(v,r)=\{u\in V(G)\colon \mathrm{dist}_G(v,u)\le r\}$。类似地，我们将以 v为中心、半径为 r的入球记为 $B^{in}_G(v,r)=\{u\in V(G)\colon \mathrm{dist}_G(u,v)\le r\}$。此外，我们写作 $\partial B^{out}_G(v,r)=\{(u,w)\in E\colon u\in B^{out}(v,r)\wedge w\notin B^{out}(v,r)\}$来表示出球的边界边，并写作 $\partial B^{in}_G(v,r)=\{(u,w)\in E\colon u\notin B^{in}(v,r)\wedge w\in B^{in}(v,r)\}$表示入球。

在所有这些记号中，如果上下文明确，我们有时会省略下标 G 。

单源最短路径（*SSSP*）问题是：对于指定的源顶点 $s \in V(G)$，计算其到所有其他顶点 $v \in V(G)$ 的距离 $\text{dist}_G(s, v)$。当 G 不包含负环时，这等价于从 s 计算一棵最短路径树（也就是说，在这棵树中，每条从 s 到 v 的路径都是从 G出发的最短路径）。对于具有非负边权的图，Dijkstra 算法的经典算法可在近线性时间内解决单源最短路径问题。我们采用Thorup 的以下结果，该结果将 $\log n$ 开销替换为 $\log\log n$（在 RAM 模型中，详见下文关于机器模型的段落）。

▶ **引理 14** (Dijkstra 算法, [23, 57])。在任意具有非负边权的有向图 G 中，单源最短路径问题可在时间 $O(m + n\log\log n)$内求解。

▶ **引理15**（贝尔曼-福特算法， [54, 26, 9, 46]）。在任意有向图 G 中，单源最短路径问题可在时间 $O（mn）$ 内求解。

势函数 令 G 为有向图。我们将函数 $\phi : V(G) \to \mathbf{Z}$ 称为势函数。我们记 G_ϕ 为由 G 得到的图，将边权改为 $w_\phi(u, v) = w(u, v) + \phi(u) - \phi(v)$。

▶ **定义 9** (等价图)。我们称同一组顶点和边上的两个图 G、G' 为等价图，当且仅当：*(1)* G 中的任意最短路径也是 G'中的最短路径，反之亦然；*(2)* 对于任意环 C、$w_G(C) = w_{G'}(C)$。

▶**引理 16** (Johnson 技巧， [35])。令 G 为有向图，并令 ϕ 为任意势函数。则 $w_\phi(P) = w(P)+\phi(u)-\phi(v)$ 对于任意 u-v-路径 P，以及 $w_\phi(C) = w(C)$对于任意环 C。由此可知， G 和 G_ϕ 等价。

▶**引理 17** （[35]）。设 G 是一个无负环的有向图，且 $s \in V$是一个能够到达其他每个节点的源顶点。则对于定义为 $\phi\ \phi（v）= \text{dist}_G（s, v）$ 的势函数，有 $w_\phi（e）\geq 0$ 对所有边 $e \in E$ 均成立。

机器模型 我们使用标准的字 RAM 模型，字长为 $\Theta（\log n+\log M）$，其中 n 是顶点数量， M 是最大边权绝对值的上界。也就是说，我们假设可以将顶点标识符和边权存储在一个机器字中，并在单位时间内执行基本操作。

3 受限图上的单源最短路径

本节将给出一种在受限图上求解单源最短路径的高效算法（回顾定义 6）。具体而言，我们将证明以下定理：

▶ **定理 18** （受限单源最短路径）。 对于带有源顶点 $s \in V（G）$ 的受限图 G，我们可以在 s 时间内计算从 $O（m + n\log\log n）\log^2 n$ 的最短路径树，错误概率为常数 $1\ 2$。（如果算法未能成功，则返回失败。）

我们分两步开发这一算法：首先，我们证明适用于受限图的分解方案（第 3.1 节），然后利用该分解方案构建适用于受限图的单源最短路径算法（第 3.2 节）。

算法 1 图分解。此算法 Decompose(G,κ) 计算边的一个子集 $S \subseteq E(G)$，使其满足引理 8 中的性质。

过程 DECOMPOSE(G,κ)
  令 $L^{in} \subseteq V(G)$ 为引理19 在 G 上标记为入轻的顶点
  令 $L^{out} \subseteq V(G)$ 为引理19 在 G^{rev} 上标记为出轻的顶点
  $S \leftarrow \emptyset$
  当 存在 $v \in V(G) \cap L^{out}$ **时**
    采样 $r \sim$ 几何分布(20 log n/κ)
    $S \leftarrow S \cup \partial B^{out}_{G_{\geq 0}}(v,\ r)$
    $G \leftarrow G \setminus B^{out}_{G_{\geq 0}}(v,r)$
  当存在 $v \in V(G) \cap L^{in}$ **时**
    示例 $r \sim$ 几何分布(20 log n/κ)
    $S \leftarrow S \cup \partial B^{in}_{G_{\geq 0}}(v,\ r)$
    $G \leftarrow G \setminus B^{in}_{G_{\geq 0}}(v,r)$
  返回 S

3.1 受限图的分解

本节中，我们证明分解引理:

▶ **引理 8** (分解). 设 G 是一个源顶点为 $s \in V(G)$ 且$\kappa \geq \kappa(G)$的受限图。存在一个随机化算法 *Decompose*(G,κ)，其期望运行时间为$O((m+n\log\log n)\log n)$，可计算出一个边集 $S \subseteq E(G)$，满足:

1. 进展：以高概率，对于 $G\setminus S$ 中的任意强连通分量 C，我们都有 *(i)* $|C| \leq \frac{3}{4}|V(G)|$ 或 *(ii)* $\kappa(G[C \cup \{s\}]) \leq \frac{\kappa}{2}$。*2.* 稀疏命中：对于 G 中任意最短的 s-v-路径 P，都有 $E(|P \cap S|) \leq O(\log n)$。

为证明该结论，我们引入一些记号。令 $G_{\geq 0}$ 表示将图 G 中的负边权替换为 0 后得到的图。若 $|B^{out}_{G_{\geq 0}}(v, \frac{\kappa}{4})| > \frac{n}{2}$，则称顶点 v 为出重 顶点；若 $|B^{out}_{G_{\geq 0}}(v, \frac{\kappa}{4})| \leq \frac{3n}{4}$，则称其为出轻 顶点。注意，有些顶点可能既是出重顶点，又是出轻顶点。类似地，将“$B^{in}_{G_{\geq 0}}$”替换为“$B^{out}_{G_{\geq 0}}$”，即可定义入轻 和入重 顶点。

▶ **引理19** (重轻分类)。 存在一种算法，给定有向图 G，可将每个顶点正确标记为入轻或入重（同时属于入轻和入重的顶点可任选其一标记）。该算法的运行时间为 $O((m+n\log\log n)\log n)$，且以高概率成功。

注意，将边的方向反转后，对所得图 G^{rev} 应用此引理，我们同样可以将顶点分为出轻和出重。我们暂且省略证明，因为由第 7 节陈述并证明的引理 52 可轻易推出。

现在可以陈述分解算法了：首先，使用前一个引理将每个顶点分别标记为出轻或出重，以及入轻或入重。然后，只要 G 中存在一个被标记为出轻或入轻的顶点 v（假设它是出轻），我们就会在 v 周围切出一个球。为此，我们从几何分布 Geom(20 log n/κ) 中采样一个半径 r，切断边 $\partial B^{out}_{G_{\geq 0}}(v,\ r)$（也就是离开 $B^{out}_{G_{\geq 0}}(v,\ r)$ 的边集），并从图中移除 $B^{out}_{G_{\geq 0}}(v,\ r)$ 内的所有顶点。我们在算法 1 中总结了这一过程。接下来，我们将证明该算法的正确性。

▶**引理 20** (算法 1 的稀疏命中)。 设 P 是 G 中一条最短的 s-v-路径，且设 S 为 *Decompose*(G,κ) 的输出。那么 $\mathbf{E}(|P \cap S|) \leq O(\log n)$。

证明。 关注任意一条边 $e = (x, y) \in E(G)$。我们分析 $e \in S$ 的概率。我们首先分析 e 被加入 S 在第7 行（边在第 11行被加入的情况也适用相同的分析）。关注第 5 行循环中针对某个出轻顶点 v 的任意一次迭代。共有三种情况：

- $x, y \notin B^{out}_{G_{\geq 0}}(v, r)$：边 e 在本次迭代中未被触及。它可能会，也可能不会在后续迭代中被纳入。
- $x \in B^{out}_{G_{\geq 0}}(v, r)$ 和 $y \notin B^{out}_{G_{\geq 0}}(v, r)$：边 e 包含在 $\partial B^{out}_{G_{\geq 0}}(v, r)$ 中，因此一定会被纳入 S。
- $y \in B^{out}_{G_{\geq 0}}(v, r)$：边 e 肯定不会被纳入 S。事实上，$e \notin \partial B^{out}_{G_{\geq 0}}(v, r)$，因此我们不会在本次迭代中将 e 纳入 S。此外，由于本次迭代后我们会从 G 中移除 y，因此之后将不再考虑边 e。

回想一下，半径 r 是从几何分布 $\mathrm{Geom}(p)$ 中抽样得到的，其中 $p := 20 \log n/\kappa$。因此，我们有

$$\begin{aligned}
\mathbf{P}(e \in S) &\leq \max_{v \in V} \mathop{\mathbf{P}}_{r \sim \mathrm{Geom}(p)}(y \notin B^{out}_{G_{\geq 0}}(v, r) \mid x \in B^{out}_{G_{\geq 0}}(v, r)) \\
&= \max_{v \in V} \mathop{\mathbf{P}}_{r \sim \mathrm{Geom}(p)}(r < \mathrm{dist}_{G_{\geq 0}}(v, y) \mid r \geq \mathrm{dist}_{G_{\geq 0}}(v, x)) \\
&\leq \max_{v \in V} \mathop{\mathbf{P}}_{r \sim \mathrm{Geom}(p)}(r < \mathrm{dist}_{G_{\geq 0}}(v, x) + w_{G_{\geq 0}}(e) \mid r \geq \mathrm{dist}_{G_{\geq 0}}(v, x))
\end{aligned}$$

根据几何分布的无记忆性，我们可以将 r 替换为（非负）随机变量 $r' := r - \mathrm{dist}^{G_{\geq 0}}(v, x)$：

$$\begin{aligned}
&= \max_{v \in V} \mathop{\mathbf{P}}_{r' \sim \mathrm{Geom}(p)}(r' < w_{G_{\geq 0}}(e)) \\
&= \mathop{\mathbf{P}}_{r' \sim \mathrm{Geom}(p)}(r' < w_{G_{\geq 0}}(e)) \\
&\leq p \cdot w_{G_{\geq 0}}(e).
\end{aligned}$$

最后一个不等式成立，因为我们可以将 $r' \sim \mathrm{Geom}(p)$ 解释为直到得到正面所需的抛硬币次数，其中每次抛掷相互独立，且出现正面的概率为 p。因此，根据并集界，$\mathbf{P}(r' < w_{G_{\geq 0}}(e))$ 至多为 $w_{G_{\geq 0}}(e)$ 次抛硬币中至少有一次出现正面的概率。

现在考虑最短的 s-v-路径 P，其位于 G 中。回顾一下，$w_G(P) \leq 0$，因为 G 是一个受限图。因此，P 至多包含 $\kappa(G) \leq \kappa$ 条负权边（即权重恰好为 -1 的边）。由此可得 $w_{G_{\geq 0}}(P) \leq \kappa$，最终：

$$\mathbf{E}(|P \cap S|) = \sum_{e \in P} \mathbf{P}(e \in S) = \sum_{e \in P} p \cdot w_{G_{\geq 0}}(e) \leq p \cdot w_{G_{\geq 0}}(P) \leq p\kappa = O(\log n). \qquad \blacktriangleleft$$

下文中，我们将需要以下引理。

▶ **引理 21。** 令 G 为一个有向图。则 $\min_C \bar{w}(C) = \min_Z \bar{w}(Z)$ 其中 C 遍历所有环，Z 遍历 G中的所有闭游走 。

证明。记 $c = \min_C \bar{w}(C)$ 和 $z = \min_Z \bar{w}(Z)$ 为 $c \leq z$。取边数最少的闭游走 Z，它见证了 z。若 Z 是一个环，则显然有 $c \leq z$。否则，Z 必定至少重访一个顶点，因此可拆分为两个闭游走 Z_1、Z_2。根据 Z 的最小性，我们有 $\bar{w}(Z_1), \bar{w}(Z_2) > z$。但请注意

$$z \cdot |Z| = w(Z) = w(Z_1) + w(Z_2) > z \cdot |Z_1| + z \cdot |Z_2| = z \cdot |Z|,$$

矛盾。 $\blacktriangleleft$

▶ **引理 22** (算法 1 的进展)。 令 S 为 *Decompose*(G,κ)的输出。那么，以高概率， $G\setminus S$ 中的任意强连通分量 C都满足 ***(i)*** $|C|\le\frac{3}{4}|V(G)|$或 ***(ii)*** $\kappa(G[C])\le\frac{\kappa}{2}$。

证明。在全文中，条件限定于重轻分类成功的事件（这以高概率发生）。注意，每当我们刻出一个球 $B^{out}_{G_{\ge0}}(v,r)$，并将其出边 $\partial B^{out}_{G_{\ge0}}(v,r)$ 纳入 S 时，任意两个顶点 $x\in B^{out}_{G_{\ge0}}(v,r)$ and$y\notin B^{out}_{G_{\ge0}}(v,r)$ 都不可能属于 $G\setminus S$ 中的同一个强连通分量（因为不存在从 x to y 的路径）。同样的论证适用于 $B^{in}_{G_{\ge0}}(v,r)$。

因此，强连通分量只有两种类型：(i) 包含在 $B^{out}_{G_{\ge0}}(v,\ r)$ 或 $B^{in}_{G_{\ge0}}(v,\ r)$ 中的分量；(ii) 在移除所有轻顶点后剩余图中的分量。我们论证，类型 (i) 的每个分量都满足 $|C|\le\frac{3}{4}|V(G)|$（以高概率），而类型 (ii) 的每个分量都满足 $\kappa(G[C])\le\frac{\kappa}{2}$。

在情况 (i) 中，我们有 $|C|\le|B^{out}_{G_{\ge0}}(v,\ r)|$。由于 v 是出轻的，因此只要 $r\le\frac{\kappa}{4}$，就有 $|C|\le\frac{3}{4}|V(G)|$。此事件以高概率发生，因为：

$$\mathbf{P}_{r\sim\mathrm{Geom}(20\log n/\kappa)}\left(r>\frac{\kappa}{4}\right)\le\left(1-\frac{20\log n}{\kappa}\right)^{\frac{\kappa}{4}}\le\exp(-5\log n)\le n^{-5}.$$

迭代次数上界为 n，因此由并集界可知，我们以至少 $1-n^{-4}$ 的概率始终不会有 $r>\frac{\kappa}{4}$。当 v 处于入轻状态时，如果我们对 $B^{in}_{G_{\ge0}}(v,\ r)$ 进行切分，类似的论证同样适用。

接下来，聚焦于情形(ii)。设 C 是从剩余图 G 中刨除所有以轻顶点为中心的球后得到的一个强连通分量。假设 $\kappa(G[C])>\frac{\kappa}{2}$。我们将构造一个在 G 中的闭游走 Z，其平均权重为 $\bar{w}(Z)<1$，从而与 G 受引理 21 限制这一假设矛盾。令 P为 $G[C\cup\{s\}]$中的 s-v-路径，该路径具有非正权重，并且见证了最大的负权边数量（即，见证 $\kappa(G[C\cup\{s\}])$ 的路径），再令 u为该路径 P 上（在 s 之后）的第一个顶点。令 P_1 为从 P 中移除 s-u-边后得到的 u-v-路径。由于 s-u-边的权重为 0，我们有 $w(P_1)\le0$，并且 P_1 包含多于 $\frac{\kappa}{2}$ 条负权边。由于 u、v 在原始图 G 中都是出重且入重顶点，我们有$|B^{out}_{G_{\ge0}}(v,\frac{\kappa}{4})|$、$|B^{in}_{G_{\ge0}}(u,\frac{\kappa}{4})|>\frac{n}{2}$。由此可知，这两个球必相交，并且存在一条 v-u-路径 P_2，其长度为 $w(P_2)\le\frac{\kappa}{4}+\frac{\kappa}{4}=\frac{\kappa}{2}$。结合 P_1 和 P_2，我们得到一个闭游走 Z，其总权重为 $w(Z)\le\frac{\kappa}{2}$，并且包含多于 $\frac{\kappa}{2}$ 条（负权）边。因此，$\bar{w}(Z)<1$，从而得到所声称的矛盾。◀

引理8的证明。 正确性由前述引理立即得到：引理 22 证明了进展性质，而引理 20 证明了稀疏命中性质。下面分析运行时间。计算重轻分类所需时间为 $O((m+n\log\log n)\log n)$，这是由于引理19。根据几何分布 r Geom(20 log n/κ)对每个半径进行采样，在字长为 Ω(log n) [13], 的字RAM 上的期望运行时间为常数，因此采样半径的开销在期望意义下为 $O(n)$。计算球时，我们使用 Dijkstra 算法。使用 Thorup 优先队列时，Dijkstra 算法每探索一个顶点需要 [57], 时间 $O(\log\log n)$，每处理一条边需要 $O(1)$。由于包含在某个球中的每个顶点都会从后续迭代中移除，因此每个顶点至多参与一个球。注意，直接实现这一过程会在 while 循环的每次迭代中重新初始化优先队列和距离数组。为避免这一点，我们在 while 循环执行之前只初始化一次优先队列和距离数组。随后，在while 循环的一次迭代结束时，我们用与被移除的顶点和边数成正比的时间重新初始化它们

（这与BNW算法 [11] 中采用的方法相同）。因此，计算所有球的总时间确实为 $O(m + n \log\log n)$。◀

3.2 定理 18 的证明

有了图分解，我们就可以给出受限单源最短路径的完整算法。整体结构与 BNW 算法非常相似（见 [11, 算法 1]）。

我们先给出以下关键定义。

▶**定义 23.** 设 G 为一个有向图，其指定源顶点为 s。对于任意顶点$v \in V(G)$，我们用 $\eta_G(v)$ 表示任意一条最短 s-v- 路径中负权边的最少数量。

下一个命题描述了参数 $\kappa(G)$ 和 $\eta_G(\cdot)$ 之间的关系，其中 G 受到限制（见定义 7 和 23）。

▶**命题 24.**令 G 为一个受限图，其源顶点为 s。那么，对于每个顶点$v \in V$，都有 $\eta_G(v) \leq \kappa(G)$.

证明。固定一个顶点 v。令 P 为一条见证 $\eta_G(v)$的最短 s-v 路径（见定义 23）。由于 G 受到限制，它不包含负环，因此 P 是一条简单路径。此外，由于存在一条从 s 到 v 、权重为 0的边，因此可得 $w_G(P) \leq 0$。回顾一下， $\kappa(G)$ 是从 s 开始且具有非正权重的任意简单路径中负权边的最大数量（见定义 7）。因此，可得 $\eta_G(v) \leq \kappa(G)$。◀

接下来，我们使用来自 [11] 的两个引理：

▶ **引理 25** （带负权的Dijkstra算法，类似于 [11, 引理 3。3]）。 设 G 为一个有向图，其源顶点为 $s \in V(G)$ ，且不包含负环。存在一种算法，可从 s 计算最短路径树，时间为 $O(\sum_v(\deg(v) + \log\log n) \cdot \eta_G(v))$。 （如果 G 包含负环，该算法不会终止。）

与 [11, 引理 3.3] 的主要区别在于，我们为 Dijkstra 算法使用了更快的优先队列，并且 [11, 引理 3.3] 仅适用于最大度数为常数的图。因此，我们在附录 A 中给出引理 25 的完整证明。

▶ **引理 26** (DAG边， [11, 引理 3.2])。 设 G 为一个有向图，其强连通分量内部的边权非负。那么，我们可以在时间 $O(n + m)$内计算出一个势函数 ϕ ，使得 G_ϕ 处处具有非负边权。

证明概要。 完整证明见 [11, 引理 3.2]。其思路是将图视为由强连通分量构成的有向无环图，并为每个强连通分量 ϕ 赋予一个势函数，使 DAG 边的权重变为非负。一种实现方法是计算一个拓扑序，并将 $\phi(v)$ 设为 W 乘以 v在该拓扑序中的强连通分量排名（这里， $-W$ 是 G中的最小权重）。那么， G_ϕ 满足所述结论。◀

算法现在我们可以陈述该算法了；伪代码见算法 2。回顾一下， $\kappa(G)$ 是以 s 开始的任意路径 P 中负权边的最大数量，其中 $w(P) \leq 0$ （定义 7）。如果 $\kappa(G) \leq 2$，我们运行引理 25 来计算从 s 出发的距离。否则，我们首先应用图分解。也就是说，我们计算一个边集 S，使得图 $G \setminus S$ 中的任意强连通分量 C 要么规模较小，要么具有改进的 κ 值。这足以递归地解决诱导子图 $G[C \cup \{s\}]$。递归调用会生成最短路径树，从而得到一个

算法 2 在受限图上求解负权单源最短路径问题。过程 RestrictedSSSP(G, κ) 接收一个受限图 G 和一个上界 $\kappa \geq \kappa(G)$，并计算从指定源顶点 s出发的最短路径树。

```
1  过程 RestrictedSSSP(G, κ)
2      如果 κ ≤ 2，则
3          对 G 从 s 运行引理 25，并返回计算得到的最短路径树
4      计算 S ← Decompose(G, κ)（参见引理 8）
5      计算 C_1, ..., C_ℓ 的强连通分量 G \ S（见引理 12）
6      对于 i ← 1, ..., ℓ do
7          如果 |C_i| ≤ 3n/4，则 κ_i ← κ，否则 κ_i ← κ/2
8          递归调用 RestrictedSSSP(G[C_i ∪ {s}], κ_i)
9          令 φ_1(v) = dist_{G[C_i ∪ {s}]}(s, v) 对所有 v ∈ C_i
10     在 (G \ S)^{φ_1} 上运行引理 26，得到一个势函数 φ_2
11     在 G_{φ_2} 上运行引理 25，并返回计算得到的最短路径树
```

势函数 ϕ_1，使得 G_{ϕ_1} 在每个强连通分量内部的边权均非负。随后，我们通过先调用引理 26（修复强连通分量之间的边 $e \notin S$），再调用引理 25（修复边 $e \in S$），将缺失的边加回。正确性证明很简单：

▶**引理 27**（算法 2 的正确性）。设 G 是一个任意的有向图（不一定是受限图），且 κ 是任意的。那么，如果 *RestrictedSSSP*（G, κ）终止，它会正确地计算出从指定源顶点 s出发的最短路径树。

证明。如果 $\kappa \leq 2$ 且第 3 行中的调用终止，那么根据引理 25，它会正确地计算出最短路径树。如果 $\kappa > 2$，那么在第 10 行中，我们计算一个势函数 ϕ_2，并在第 11 行中运行引理 25，在图 G_{ϕ_2}中计算最短路径树。假设引理 25 终止，这一计算是正确的，因为G_{ϕ_2} 等价于 G。◀

▶ **引理 28**（算法 2 的运行时间）。设 G 是一个满足 $\kappa(G) \leq \kappa$的受限图。那么，*RestrictedSSSP*（G, κ）的期望运行时间为 $O((m + n\log\log n)\log^2 n)$。

证明。我们首先分析对算法 2 的单次调用的运行时间，不计递归调用所花费的时间。对于基本情况，当 $\kappa(G) \leq 2$时，根据引理 25 和命题 24，第 3 行的运行时间为 $O(m + n\log\log n)$。否则，根据引理 8，第 4 行对$\textsc{Decompose}(G, \kappa)$的调用耗时 $O((m + n\log\log n)\log n)$。在 $G \setminus S$ 中计算强连通分量的时间为线性时间 $O(m + n)$，第 10 行对引理 26 的调用也是如此。

分析第11 行的运行时间还需要一些额外的工作。回想一下，$\eta_{G_{\phi_2}}(v)$ 是任意 s-v 路径中负权边的最小数量，其中路径位于 G_{ϕ_2}（见定义 23）。我们的中间目标是：对于所有顶点 v，界定 $\mathbf{E}(\eta_{G_{\phi_2}}(v)) = O(\log n)$。令 S 为算法中通过分解计算得到的边集。我们分三步进行：

- 断言 *1*：$G_{\phi_1} \setminus S$ 其强连通分量内部的边权均非负。根据引理 27，第 8 行的递归调用正确计算了距离。因此，对于任意两个节点 $u, v \in C_i$，我们有 $w_{\phi_1}(u, v) = w(u, v) + \text{dist}_{G[C_i \cup \{s\}]}(s, u) - \text{dist}_{G[C_i \cup \{s\}]}(s, v) \geq 0$，由三角不等式可得。断言 *2*：$G_{\phi_2} \setminus S$ 仅包含非负权边。由引理 26 可直接得出。

主张 3：对于每个节点 v，我们有 $\mathbf{E}(\eta_{G_{\phi 2}}(v)) \leq O(\log n)$。令 P 为 G 中一条最短的 s-v-路径。由于 G 和 G_{ϕ_2} 等价，P 也是 G_{ϕ_2} 中的一条最短路径。根据前一个主张，P 中唯一可能的负权边是 S 中的边。因此，根据引理 8，我们有 $\mathbf{E}(\eta_{G_{\phi 2}}(v)) \leq \mathbf{E}(|P \cap S|) = O(\log n)$。

因此，第 11 行的预期运行时间有界于

$$\begin{aligned} & O\left(\sum_{v \in V(G)} (\deg(v) + \log\log n) \cdot \mathbf{E}(\eta_{G_{\phi_2}}(v))\right) \\ &= O\left(\sum_{v \in V(G)} (\deg(v) + \log\log n) \cdot \log n\right) \\ &= O((m + n\log\log n)\log n). \end{aligned}$$

因此，算法2 的一次执行时间为 $O((m + n\log\log n)\log n)$；令 c 表示 O 表示法中的隐藏常数。

最后，我们分析总运行时间，并将递归调用考虑在内。我们通过归纳证明，运行时间有界于 $c(m + n\log\log n)\log n \cdot \log^{4/3}(n\kappa)$。

我们断言，对于在子图 $G[C_i \cup \{s\}]$, 上的每次递归调用，其中 C_i 是 $G \setminus S$ 中的一个强连通分量，都有：(i) $G[C_i \cup \{s\}]$ 是一个受限图，并且 (ii) $\kappa(G[C_i \cup \{s\}]) \leq \kappa_i$。为说明(i)，注意到，$G$ 的任何包含 s 的子图也是受限图。为证明 (ii)，我们区分两种情况：要么 $|C_i| \leq \frac{3n}{4}$，在这种情况下，我们显然有 $\kappa(G[C_i \cup \{s\}]) \leq \kappa(G) \leq \kappa = \kappa_i$。要么 $|C_i| > \frac{3n}{4}$，在这种情况下，引理8 保证 $\kappa(G[C_i \cup \{s\}]) \leq \frac{\kappa}{2} = \kappa_i$。由归纳法可知，每次递归调用的运行时间为 $c \cdot (|E(G[C_i \cup \{s\}])| + |C_i|\log\log n)\log n \cdot \log^{4/3}(|C_i|\kappa_i)$。此外，注意在这两种情况下我们都有 $|C_i|\kappa_i \leq \frac{3}{4}n\kappa$。因此，总时间可以界定为

$$\begin{aligned} & c(m + n\log\log n)\log n + \sum_{i=1}^{\ell} c \cdot (|E(G[C_i \cup \{s\}])| + |C_i|\log\log n)\log n \cdot \log_{4/3}(|C_i|\kappa_i) \\ &\quad \leq c(m + n\log\log n)\log n \\ &\qquad + \sum_{i=1}^{\ell} c \cdot (|E(G[C_i \cup \{s\}])| + |C_i|\log\log n)\log n \cdot (\log_{4/3}(n\kappa) - 1) \\ &\quad \leq c(m + n\log\log n)\log n + c(m + n\log\log n)\log n \cdot (\log_{4/3}(n\kappa) - 1) \\ &\quad = cm\log n\log\log n \cdot \log_{4/3}(n\kappa), \end{aligned}$$

其中，在第三步中，我们使用了 $\sum_i |E(G[C_i \cup \{s\}])| \leq m$ 以及 $\sum_i |C_i| \leq n$。至此，运行时间分析完成。◀

定理18 的证明。 该证明几乎可以直接由前面的两个引理 27和28 得出。结合这两个引理可证明，算法2 是一个用于受限单源最短路径问题的拉斯维加斯算法，其期望时间为 $O((m + n\log\log n)\log^2 n)$。在算法运行时间达到其期望运行时间两倍后中断算法（并在此情况下返回失败），我们便得到一个最坏情况运行时间为 $O((m + n\log\log n)\log^2 n)$ 且错误概率为 $\frac{1}{2}$ 的蒙特卡洛算法，正如所声称的那样。◀

我们指出，即使输入图 G 不受限制，算法 2 仍然正确——因此，只要 G 包含负环，该算法就无法终止。

算法 3 缩放算法的一步。给定一个最小权重大于 $-3W$ 的图 G，Scale(G) 会计算一个势函数 ϕ，使得 G_ϕ 的最小权重大于 $-2W$。参见引理 30。

```
过程 SCALE(G)
    令 W，使得 G 中的所有权重都大于 −3W
    令 H 是 G 的一个副本，其边权为 w_H(e) = ⌈w_G(e)/W⌉ + 1，并添加
      一个人工源顶点 s，通过权重为 0 的边连接到所有其他顶点 H
    使用定理 18 从 s 计算受限图 H 的最短路径树
    令 φ 为由 φ(v) = W · dist_H(s, v) 定义的势函数
    返回 φ
```

4 无负环图上的单源最短路径

在本节中，我们介绍一种用于不含负环的图上求解单源最短路径的 $O((m+n\log\log n)\log^2(n)\log(nW))$-时间算法。随后在第 5节中，我们将移除 G 不含负环这一假设，并强化该算法，使其在存在负环时能够找出负环。

核心思想是使用缩放以及一些概率放大技巧，将我们在第 3 节中为受限图开发的算法加以扩展。更准确地说，我们使用标准的缩放技术 [32, 33, 34, 11]，将任意图（不含负环）上的单源最短路径计算归约为受限图的情形。形式化地，我们证明以下定理：

▶**定理 29**（单源最短路径缩放算法）。存在一种拉斯维加斯算法，给定一个不含负环的有向图 G 以及源顶点 $s \in V(G)$，能够从 s 计算最短路径树，并以 $O(T_{RSSSP}(m,n)\cdot\log(nW))$ 的时间运行，且以高概率（以及在期望意义下）成立。

单步缩放 缩放算法的思路是在 G中逐步增加最小权重，同时维护一个等价图。下面的引理 30给出了将此类缩放步骤实现为到受限单源最短路径的直接归约。

▶**引理 30** (单步缩放)。设 G 是一个不包含负环且最小权重大于 $-3W$ 的有向图（对于某个整数 $W \geq 1$）。存在一个算法 *Scale*(G)，用于计算 ϕ，使得 G_ϕ 的最小权重大于 $-2W$；该算法以恒定概率成功（如果算法失败，则返回 *Fail*），并在 $O(T_{RSSSP}(m,n))$时间内运行。

证明。我们构造一个受限图 H，它是 G 的副本，边权 $w_H(e) = \lceil w_G(e)/W \rceil + 1$ 经过修改。我们还向 H 添加一个源顶点 s，并添加从 s 到所有其他顶点、权重为 0 的边。我们使用定理 18 计算 H 中从 s 出发的最短路径树，并返回由 $\phi(v) = W\cdot \mathrm{dist}_H(s,v)$ 定义的势函数 ϕ。伪代码见算法 3。注意，运行时间主要由在受限图中计算最短路径所决定。

为了证明该算法正确，我们首先检查 H 确实是受限的（见定义 6）：

- 每条边的权重满足 $w_H(e) = \lceil w_G(e)/W \rceil + 1 \geq \lceil (-3W+1)/W \rceil + 1 = -1$。

18 近线性时间求解带负权单源最短路径：更快了！

算法 4 快速单源最短路径算法。对于给定的图 G （不含负环），该算法从给定的源顶点 s出发，在 G 中计算一棵最短路径树。

1 过程 单源最短路径(G, s)**2** 令 $-W$ 为 G 中最小的边权**3** 令 G_0 为 G 的副本，其边权为 $w_{G_0}(e) = 4n \cdot w_G(e)$**4 对于** $i \leftarrow 0, \ldots\ldots, L-1$，其中 $L = \Theta(\log(nW))$ **执行 5** $\phi_i \leftarrow$Scale(G_i)（重复运行该算法，直至成功）**6** $G_{i+1} \leftarrow (G_i)_{\phi_i}$**7** 令 G^* 为 G_L 的副本，并将负权替换为 0**8 返回** 在 G^* 中从 s 出发、由 Dijkstra 算法计算得到的最短路径树

- 考虑 H中的任意一个环 C。回想一下， $w_G(C) \geq 0$ （因为 G 不包含负环），因此

$$\bar{w}_H(C) = \frac{w_H(C)}{|C|} = \frac{1}{|C|}\sum_{e\in C} w_H(e) = 1 + \frac{1}{|C|}\sum_{e\in C}\left\lceil w_G(e)\cdot\frac{1}{W}\right\rceil \geq 1 + \frac{w_G(C)}{W|C|} \geq 1.$$

  特别地， H 中的最小环均值至少为 1。
- 最后，我们人为地向 H 中添加了一个源顶点 s，并添加权重为 0 的边，使其连接到所有其他顶点。

仍需证明，由 $\phi(v) = W\cdot \text{dist}_H(s,v)$ 定义的势函数 ϕ 满足： G_ϕ 的最小边权大于 $-2W$。考虑任意一条边 $e=(u,v)$。由于根据定义， $w_H(e) < w_G(e)\cdot\frac{1}{W}+2$，因此我们有 $w_G(e) > W\cdot(w_H(e)-2)$。由此可得

$$\begin{aligned} w_{G_\phi}(e) &= w_G(e) + \phi(u) - \phi(v) \\ &= w_G(e) + W\cdot\text{dist}_H(s,u) - W\cdot\text{dist}_H(s,v) \\ &> -2W + W\cdot w_H(e) + W\cdot\text{dist}_H(s,u) - W\cdot\text{dist}_H(s,v) \\ &\geq -2W. \end{aligned}$$

在最后一步中，我们使用了三角不等式 $\text{dist}_H(s,v) \leq \text{dist}_H(s,u) + w_H(u,v)$。

最后，我们论证该算法以常数概率成功。注意，如果第4 行中从 s 计算最短路径树成功，则该算法成功（事实上，所有其他步骤都是确定性的）。由于 H 受到限制，定理 18保证这一点以常数概率成立；如果不成立，则返回失败，从而完成证明。◀

完整缩放算法 我们现在可以陈述算法 SSSP(G,s)，它实现了定理 29。我们将 G 的每条边权乘以 $4n$，构造图 G_0。然后，对于 $i=0,\ldots,L-1$，其中 $L=\Theta(\log(nW))$，我们调用 Scale(G_i)（重复调用直到成功），以获得势函数 ϕ_i，并设置 $G_{i+1} := (G_i)_{\phi_i}$。接下来，我们构造图 G^*，它是 G_L 的副本，其中每条负边权都替换为 0。最后，我们使用 Dijkstra 算法在 G^* 中计算一棵最短路径树。详情请参阅算法 4 中的伪代码。

▶**引理 31**（算法 4 的运行时间）。如果 G 不包含负环，那么单源最短路径（G, s）以 O（T_{RSSSP}（m, n）·log（nW）的运行时间高概率（且以期望意义）完成。

证明。我们分析 for 循环的运行时间，它会运行 $L = O(\log(nW))$ 次迭代。每次迭代都会重复调用 $\textsc{Scale}(G_i)$，直到其中一次调用成功。根据引理 30，a

单次调用以常数概率（例如 1/2）成功，运行时间为 $O(T_{\mathrm{RSSSP}}(m,n))$。因此，我们可以将第 i 次迭代的运行时间建模为 $O(X_i \cdot T_{\mathrm{RSSSP}}(m,n))$，其中 $X_i \sim \mathrm{Geom}(1/2)$ 是一个几何分布随机变量。因此，根据 Chernoff 界，for 循环的运行时间以至少 $1-\exp(-\Omega(L)) \geq 1-n^{-\Omega(1)}$ 的概率有界于 $O(\sum_{i=0}^{L-1} X_i \cdot T_{\mathrm{RSSSP}}(m,n)) = O(T_{\mathrm{RSSSP}}(m,n) \cdot L)$。最后，注意到 $T_{\mathrm{RSSSP}}(m,n) = \Omega(m+n)$，因此第 8 行对 Dijkstra 算法的调用所需时间受 for 循环的运行时间支配。◀

▶ **引理 32**（算法 4 的正确性）。 如果 G 不包含负环，则算法 *4* 能够从 s 正确计算出一棵最短路径树。

证明。 考虑算法 4 的一次执行。我们证明 G^* 中的任意最短路径都是 G中的最短路径，因此从 s 中计算出的 G^* 最短路径树也是 s 在 G 中的最短路径树，从而证明正确性。我们分三步进行:

- 由于 G_0 是 G 的副本，其边权经过缩放，为 $w_{G_0}(e) = 4n \cdot w_G(e)$，所以任何路径 P 的缩放后权重也为 $w_{G_0}(P) = 4n \cdot w_G(P)$，因此 G 和 G_0 等价。由于图 G_0、……、G_L 是通过添加势函数相互得到的，因此它们等价（见引理 16）。此外，根据引理 30 的性质，最小权重 $-W$ 在每一步都会增大 2 3 倍，直到 G_L 的最小权重至少为 -3。这里我们利用了对于足够大的隐藏常数，$\bar{L} = \Omega(\log(nW))$。$G^*$ 是通过将 G_L 中的负权边替换为权重为 0 的边得到的图。考虑 G_L 中任意一条非最短的 u-v 路径 P'。我们将证明 P' 在 G^* 中也不是一条最短的 u-v 路径，从而完成论证。为此，设 P 是任意一条最短的 u-v 路径。回顾一下，G_L 等于 $(G_0)^\phi$，其中 ϕ 是某个势函数。因此:

$$\begin{aligned} & w_{G_L}(P') - w_{G_L}(P) \\ & \quad = w_{G_0}(P') + \phi(u) - \phi(v) - w_{G_0}(P) - \phi(u) + \phi(v) \\ & \quad = w_{G_0}(P') - w_{G_0}(P) \\ & \quad \geq 4n, \end{aligned}$$

其中，最后一个不等式利用了 G_0 中 P 和 P' 的权重至少相差 $4n$ 这一点（这就是我们为何在 G_0 中将边权缩放 $4n$ 的原因）。最后，回想一下，转移到 G^* 后，任意路径的权重最多增加 $3\cdot(n-1)$。由此可得，

$$w_{G^*}(P') - w_{G^*}(P) \geq w_{G_L}(P') - w_{G_L}(P) - 3\cdot(n-1) \geq 4n - 3\cdot(n-1) > 0,$$

因此，P' 不是一条最短 u-v-路径，在 G^*中。因此，G^* 中的最短路径也是 G_L中的最短路径，而且由于 G_L 等价于 G，它也是 G中的最短路径。◀

定理 29 的证明可通过结合引理 31 和 32 立即得到。
我们以如下引理结束本节，它将在下一节中派上用场。

▶ **引理33**。 设 G 是一个有向带权图，且 $s \in V$（G）。如果单源最短路径（G, s） 终止，则 G不包含负环。

证明。反设 G 存在一个负环 C，且 SSSP(G, s) 终止。考虑在第 4 行 for 循环的最后一次迭代中构造的图 G_L。注意，G_L 等价于 G_0，因为它是通过添加势函数得到的。观察到，C 在 G_0 和 G_L 中的权重至多为 $-4n$，因为它在 G 中为负，且我们按因子 $4n$ 进行了缩放（见引理 16）。回想一下，我们选择了$L = \Theta(\log(nW))$，其隐藏常数足够大，使得 G_L 中的最小权重至少为 -3。这意味着 G_L 中的最小环权重至少为 $-3n$，矛盾。

◀

5 寻找负环

在第 4 节中，我们开发了一种算法，能够以高概率在不含负环的图中计算最短路径树。本节中，我们将这一结果扩展为：如果存在负环，则找到它。作为预热，我们注意到，第 29 定理中开发的单源最短路径算法可用于以高概率检测负环是否存在：

▶**推论 34**。令 G 是一个有向图。则存在一个算法 *DetectNegCycle*(G)，具有以下性质：

- 如果 G 存在负环，则该算法会报告 *NegCycle*。如果 G 不存在负环，则该算法以高概率返回 *NoNegCycle*。它的运行时间为 $O(T_{RSSSP}(m,n)\log(nW))$。

证明。该算法添加一个虚拟源点 s，并用权重为 0 的边将其连接到 G 中的所有顶点，然后运行 单源最短路径(G,s)。如果算法在时间预算内完成，我们就返回 NoNegCycle；否则，我们中断计算并返回 NegCycle。根据定理 29 的保证，运行时间显然成立。

现在我们论证正确性。如果 G 不包含负环，则根据定理 29，该算法以高概率返回 NoNegCycle。如果 G 包含一个负环，则引理33 意味着单源最短路径$(G,\ s)$不会终止，因此在这种情况下我们总是报告 NegCycle。◀

寻找 负环则需要更多工作。为此，我们沿用 [11] 的思路。他们将寻找负环的问题归约为一个称为阈值的问题，接下来我们将对其进行定义。我们将使用以下记号：给定一个有向图 G 和整数 M，我们用 G^{+M} 表示将 M 加到 G 的每条边权上后得到的图。

▶**定义 35** (阈值)。给定一个有向图 G，阈值(G) 是满足 G^{+M^*} 不包含负环的最小整数 $M^* \geq 0$ 。

对于图 G，我们将 $T_{\text{THRESHOLD}}(m,\ n)$ 记为以高概率运行算法计算阈值(G) 的最优运行时间。

本节其余部分的组织如下：在第 5.1 节中，我们给出从寻找负环到阈值问题的归约。在第5.2节中，我们给出阈值问题的一种实现，该实现与承诺的定理 1 相比多出一个对数因子，但其优点是简单。最后，在第5.3节中，我们给出阈值问题的一种更快（但更复杂）的实现，从而得到定理 1。

5.1 归约到阈值

在本节中，我们重述 Bernstein 等人在 [11, 第 7 节1] 中给出的归约：将“若存在则寻找负环”的问题归约为阈值和 RestrictedSSSP（参见他们的算法 SPLasVegas）。

▶ **引理 36** （寻找负环）。设 G 是一个含有负环的有向图。存在一个拉斯维加斯算法 *FindNegCycle*(G)，它能在 G中找到一个负环，并以高概率在 $O(T_{RSSSP}(m,n)\log(nW)$ $+T_{THRESHOLD}(m,n))$ 时间内运行完毕。

算法 5 寻找负环的过程。给定一个包含负环的图 G，该过程会找到一个这样的负环 C。参见引理 36。

```
1  过程 FindNegCycle(G)
2      令 G_0 为 G 的副本，其中有 w_{G_0}(e) = (n^3 + 1) · w_G(e)
3      令 M* ← 阈值(G_0)
4      令 G_1 为 G_0^{+M*} 的副本。其中我们添加一个人工源点
           顶点 s to G_1，并添加权重为 0 的边连接到所有其他顶点
5      运行单源最短路径(G_1, s)（参见算法 4），并设置 φ(v) = dist_{G_1}(s, v)
6      令 G_2 为图 (G_1)_φ，其中我们移除所有权重大于 n 的边
7      if 存在一个环 C 在 G_2 then
8          如果 C 在 G 中为负，则返回 C
9      返回 FindNegCycle(G)（即，重新启动）
```

证明。简洁描述请参见算法 5 中的伪代码。我们首先定义一个图 G_0，它是 G 的副本，但边权乘以 n^3+1。然后，我们使用 Threshold(G_0) 计算 M^*，并令 G_1 为 $G_0^{+M^*}$。接下来，我们向 G_1 添加一个虚拟源点 s，并用权重为 0 的边将其连接到所有其他顶点，然后从 s 出发，在所得图上运行单源最短路径。随后，我们使用计算出的距离构造一个势函数 ϕ，并通过将势函数 ϕ 应用于 G_1，再移除所有权重大于 n 的边，构造图 G_2。最后，我们检查 G_2 是否包含任何环（权重不限）；如果包含，则检查它在原始图 G 中是否具有负权，并将其返回。否则，我们从头重新运行该算法。

正确性显而易见：算法终止时，显然会返回一个负环。有趣之处在于证明：高概率下，算法无需重启就能找到一个负环。第 3 行对阈值(G_0)的调用会返回最小的 $M^* \geq 0$，使得 G_0 不包含负环；这一结果以高概率成立。在这种情况下，根据定义，G_1 不包含负环，因此根据定理 29，对单源最短路径(G_1, s)的调用会正确地计算出从 s出发的最短路径树。接下来，我们以这两个事件均已发生为条件。

▷ 断言 37。它成立，即 $M^* > n^2$。

证明。设 C 是 G 中的最小（负）权重简单环。由于 $G_1 = G_0^{+M^*}$ 不包含负环，C 在 G_1 中的权重为 $0 \leq w_1(C) = w_0(C) + M^*|C|$。由注意到 $w_0(C) < -n^3$ 由于第 2 行中的缩放，以及 $|C| \leq n$因为 C 是简单环，即可得出该结论。◁

接下来，我们论证，G 中的最小权重环仍然是 G_2中的一个环；反过来，G_2 中的任何简单环都对应于 G中的一个负权环。请注意，这足以证明算法在无需重启的情况下以高概率终止。

▷ 断言38。令 C 为 G 中的最小权重简单环。那么，C 是 G_2中的一个环。

证明。首先注意到 C 在 $G_0^{+M^*}$（因此在 G_1中也）的权重至多为 n。这是因为 M^* 是使得 $G_0^{+M^*}$ 不含负环的最小整数，这意味着 $w_0(C) - |C| < 0$。其次，注意到由于第 5 行在 G_1中正确计算出了一棵最短路径树，因此 $(G_1)_\phi$ 中的边权均为非负（根据引理 17）。此外，C 在 $(G_1)_\phi$ 中的权重与在 G_1 中的相同（根据引理 16）。因此，我们得出结论：移除 $(G_1)_\phi$ 中权重大于 n 的边以得到 G_2，不会改变 C。◁

▷ 断言 39。任意环 C' 在 G_2 中具有负权在 G。

证明。 注意， $w_2(C') \le n^2$ 因为 G_2 中的每条边的权重至多为 n。此外，由于 G_2是通过添加一个势函数从 G_1 得到的，因此有 $w_2(C') = w_1(C')$ (根据引理 16)。因此， $w_0(C') = w_1(C') - M^*|C'| \le n^2 - M^* < 0$其中，最后一个不等式成立是因为$M^* > n^2$ 根据断言 37。 ◁

最后，我们分析运行时间。调用 Threshold(G_0) 以高概率成功（见定义 35）。在此条件下， G_1 不含负环。因此，根据定理 29，调用 SSSP(G， s) 的运行时间为 $O(T_{\text{RSSSP}}(m,\ n)\log(nW))$，且该结论以高概率成立。注意，算法的其余步骤耗时 $O(m)$。因此，我们得出总体运行时间为 $O(T_{\text{RSSSP}}(m,\ n)\log(nW) + T_{\text{THRESHOLD}}(m,\ n))$，且该结论以高概率成立。◀

5.2 阈值的简单实现

本节给出阈值的一个简单实现，结合引理 36，可得到一个在 $O(T_{\text{RSSSP}}(m,n)\log\ n\log(nW))$ 时间内寻找负环的算法。与 [11] 相比（参见他们在引理 7.2 中的算法 FindThresh ），该过程减少了一个对数因子。随后，在第 5.3 节中，我们将给出一个改进但更复杂的算法。

作为构建模块，我们将使用引理 30 中的 Scale 过程。以下引理提高了 Scale 成功的概率，并对输入图中的最小权重采用不同的参数化方式，从而使我们的表述更加简洁。

▶**引理 40** (Test Scale). 设 G 是一个有向图，其最小权重至少为 $-W$，其中 $W \ge 24$，且设 $0 < \delta < 1$ 是一个参数。存在一个算法 ***TestScale***(G, δ)具有以下性质：

- 如果 G 不包含负环，则以至少 $1-\delta$ 的概率成功，并返回一个势函数 ϕ，使得 G_ϕ 的最小权重至少为 $-\frac{3}{4}W$。如果未成功，则返回失败。其运行时间为 $O(T_{RSSSP}(m,n)\cdot \log(1/\delta))$。

证明。 我们运行 $\textsc{Scale}(G)$ （见引理 30）进行 $O(\log(1/\delta))$ 次重复。每次执行要么返回一个势函数 ϕ，要么失败。仅当所有 这些重复都失败时，我们才返回 失败。根据引理 30，运行时间的分析是直接的。

现在我们分析正确性。首先考察成功概率。引理 30 保证，如果 G 不包含负环，那么每次调用 Scale(G) 都以常数概率返回一个势函数 ϕ 。因此，在这种情况下，所有 $O(\log(1/\delta))$ 次重复都失败并返回失败的概率至多为 δ，如所述。接下来，我们分析 G_ϕ的最小权重的增加。回忆一下， G 中的最小权重至少为 $-W$。令 k 为满足 $W \ge 3k$的最大整数，并令 $-W'$ 表示 G_ϕ的最小权重。特别地， G 中的最小权重大于 $-3(k+1)$，因此引理 30 保证

$$-W' > -2(k+1) \ge -\tfrac{2}{3}W - 2 \ge -\tfrac{2}{3}W - \tfrac{1}{12}W = -\tfrac{3}{4}W,$$

其中最后一个不等式用到了假设 $W \ge 24$。 ◀

▶**引理 41** （慢速阈值）。 设 G 为一个有向图。存在一种算法计算 ***Threshold***（G）（定义 *35*），该算法以高概率成功，并且最坏情况时间为 O（T_{RSSSP}（m,n）log $n$log（nW）。

算法 6 阈值的慢速实现。给定一个图 G，它会计算最小整数 $M^* \geq 0$，使得 G^{+M^*} 不包含负环。参见引理 41。

```
过程 SlowThreshold(G)
    令 −W 为 G中的最小权重
    如果 W ≤ 48 则
        对于 t ← 47, ..., 1, 0 do
            如果DetectNegCycle(G^{+t}) = NegCycle 则返回 t + 1
        return 0
    else
        令 M ← ⌈W/2⌉
        如果 TestScale(G^{+M}, n^{-10}) = φ 则返回 SlowThreshold(G_φ)
        否则返回 M + SlowThreshold(G^{+M})
```

证明。 我们总结算法 6 中的伪代码。设 $-W$ 为 G中的最小权重。如果 $W \leq 48$（即所有权重都至少为 -48），我们显然可以确定正确答案位于范围 $0 \leq M^* \leq 48$内。我们通过穷举检查哪个图G^{+47}、.....、G^{+0} 首先包含负环，从而暴力求出答案。对于该测试，我们使用算法DetectNegCycle（G）。推论 34 保证该算法以高概率报告正确答案。

如果 $W > 48$，我们通过将问题归约为一个最小权重更大的实例来取得进展。令 $M = \lceil \frac{W}{2} \rceil$，并对 $\delta := 1/n^{10}$ 运行 TestScale(G^{+M}, δ)。我们根据 TestScale 的结果区分两种情况：

- 情况 1：TestScale(G^{+M}, δ) $= \phi$ 用于势函数 ϕ。然后递归计算并返回 SlowThreshold(G_ϕ)。首先注意这是正确的，即通过对 G_ϕ进行递归，答案不会改变，因为势函数不会改变任何环的权重（见引理 16）。其次，注意到我们通过将 G_ϕ 中的最小权重增加到至少 $-\frac{11}{12}W$来取得进展：为了解释这一点，注意到 G^{+M}的最小权重至少为 $-\frac{1}{2}W$，因此，引理 40 保证 G_ϕ^{+M}中的最小权重至少为 $-\frac{3}{8}W$。因此，可以得出 G_ϕ 中的最小权重至少为

$$-\tfrac{3}{8}W - M = \tfrac{3}{8}W - \lceil \tfrac{1}{2}W \rceil \geq -\tfrac{7}{8}W - 1 > -\tfrac{7}{8}W - \tfrac{1}{24}W = -\tfrac{11}{12}W,$$

  其中第二个不等式利用了 $W > 24$ 这一假设。
- 情况2: TestScale(G^{+M}, δ) = 失败。根据引理 40，如果 G^{+M}不包含负环，那么以高概率输出不是失败。在此事件条件下，我们得出 G^{+M} 包含负环。因此，我们知道最优答案M^*满足 $M^* \geq M$，因此返回 M + SlowThreshold(G^{+M})。注意，这还会将最负的边权改进为 $-W + M \geq -\frac{11}{12}W$。

我们声称，运行时间由 $O(T_{\mathrm{RSSSP}}(m,n) \log n \log(nW))$ 界定。要了解这一点，请注意，在基本情况中，当 $W \leq 48$时，算法调用DetectNegCycle(G)，因此耗时 $O(T_{\mathrm{RSSSP}}(m,n) \cdot \log(nW))$（见推论34）。我们声称，递归的更高层级总共耗时 $O(T_{\mathrm{RSSSP}}(m,n) \log n \log W)$。注意，由于调用 TestScale（引理 40），每一层耗时 $O(T_{\mathrm{RSSSP}}(m,n) \cdot \log n)$，因此，只需将递归深度界定为 $O(\log W)$。为此，请注意，我们总是在这样的图上进行递归：其中 W 已减少了一个常数因子。

最后请注意，每次调用 TestScale 都以高概率成功，而我们会针对 $O(\log W)$ 次递归调用中的每一次各调用一次。因此，根据并集界，该算法以高概率成功

。（严格来说，对于这个并集界，我们假设 $\log W \le n$；如果是 $\log W > n$，我们可以直接使用 Bellman-Ford 算法。）◀

5.3 阈值的快速实现

在本节中，我们给出阈值的快速实现。

▶ **引理 42** (快速阈值)。设 G 为一个有向图。存在一个算法计算阈值(G)（见定义 *35*），其以高概率成功，并在最坏情况时间 $O(T_{RSSSP}(m, n)\log(nW))$内运行。

该算法较为复杂，因此我们先进行高层次的描述，以传达一些直观理解。

高层次思路 设 Δ 为一个参数，并设 $M^* \ge 0$ 为正确阈值。我们来看看调用 TestScale($G^{+W-\Delta}, \delta$) 会发生什么，其中 $1-\delta$ 是成功概率，而 $-W$ 是 G 中的最小边权。如果 $G^{+W-\Delta}$ 不含负环，那么引理 40 保证我们以至少 $1-\delta$ 的概率得到一个势函数 ϕ。另一方面，如果 $G^{+W-\Delta}$ 含有负环，那么引理 40 对此不作保证 。也就是说，算法可能返回一个势函数，也可能返回失败。好处是，只要我们得到了一个势函数，无论是否存在负环，都可以通过（加法地）增加 $\approx \Delta$ 来推进进展。此外，如果得到失败，那么我们可以断定，图 $G^{+W-\Delta}$ 以至少 $1-\delta$ 的概率含有负环。这启发了以下思路。我们调用 TestScale($G^{+W-\Delta}, \delta$)，并考虑两种结果：

1. $\textsc{TestScale}(G^{+W-\Delta}, \delta) = \phi$。然后，我们设置 $G := G_\phi$ 并增大 $\Delta := 2\Delta$。**2.** $\textsc{TestScale}(G^{+W-\Delta}, \delta) =$失败。然后，我们减小 $\Delta := \Delta/2$。

如果处于情况 1，那么 G_ϕ 的最小边权 $-W'$ 会增加 Δ。这反过来会缩小间隙 $W' - M^*$（注意，始终有 $M^* \le W'$）。因此， Δ越大，缩小 $W' - M^*$所取得的进展就越大。这就是为什么在这种情况下我们将 Δ加倍。另一方面，如果处于情况 2，那么根据引理 40 的保证，我们可以得出结论：图 $G^{+W-\Delta}$ 以至少 $1-\delta$的概率包含一个负环。直观而言，这意味着 Δ 过大。因此，我们将 Δ 减半，以便最终再次在情况 1 中取得进展。

简而言之，我们知道，当 $G^{+W-\Delta}$ 没有负环，或等价地$W - M^* \ge \Delta$时，以至少 $1-\delta$ 的概率，我们将在情况 1 中通过减小间隙 $W - M^*$取得进展。另一方面，如果我们处于情况 2，且 $G^{+W-\Delta}$有一个负环，或等价地 $W - M^* < \Delta$，那么我们将通过减小 Δ取得进展。

或许令人惊讶的是，我们将证明，可以通过选择 $\delta = 0.01$来实现这一想法，而不是像引理 41 的实现中那样选择 $1/\text{poly}(n)$（这正是当时多出一个 $O(\log n)$-因子的原因）。为此，我们将把进展形式化为某个漂移函数，该函数在每次迭代中的期望都会减小，然后应用漂移定理（见定理 44）。

该算法现在我们将这种方法形式化。我们以迭代方式进行。在第 t 次迭代时，我们有一个图 G_t，其最小权重为 $-W_t$，并维护一个参数 Δ_t。我们调用 Scale($G^{+W_t-\Delta_t}, \delta$)，其中 $\delta := 0.01$。如果得到一个势函数 ϕ 作为答案，我们就设定 $G_{t+1} := (G_t)^\phi$ 和 $\Delta_{t+1} := 2\Delta$。否则，我们设定 $G_{t+1} := G_t$ 和 $\Delta_{t+1} := \frac{1}{2}\Delta_t$。经过 $T = \Theta(\log(nW))$ 次迭代后，我们停止并返回 W_T 作为答案。完整的伪代码（还处理了一些边界情况）见算法 7。

算法 7 阈值的快速实现。给定一个图 G，它计算最小整数 $M^* \geq 0$，使得 G^{+M^*} 不包含负环。参见引理 42。

---

1 过程 FastThreshold(G) 2 令 $G_0 \leftarrow G$ **和** $\Delta_0 \leftarrow 23$ $T \leftarrow \Theta(\log(nW))$**，其中隐藏常数足够大 4 对于** $t \leftarrow 0, \ldots, T-1$**，执行 5 令** $-W_t$ **为** G_t **中最小的边权 6 如果** $W_t \leq 24$**，则 7 对于** $j \leftarrow 23, \ldots, 1, 0$**，执行 8 如果 FindNegCycle($G_t^{+j}$)** $=$ **NegCycle，则返回** $j+1$**9 如果 TestScale($G_t^{+W_t-\Delta_t}$, 0.01)** $= \phi$**，则10** $G_{t+1} \leftarrow (G_t)^{\phi}$**，** $\Delta_{t+1} \leftarrow 2 \cdot \Delta_t$**11 否则12** $G_{t+1} \leftarrow G_t$**，** $\Delta_{t+1} \leftarrow$**max{1,** $\frac{\Delta_t}{2}$**} 13 返回** W_T

—

为量化该算法取得的进展，我们定义迭代 t时的以下 漂移函数：

$$D_t := (W_t - M^*)^{20} \cdot \max\left\{\frac{2\Delta_t}{W_t - M^*}, \frac{W_t - M^*}{2\Delta_t}\right\}, \tag{1}$$

注意，在整个算法过程中，我们始终有 $\Delta_t \geq 1$ and $W_t \geq M^*$。为涵盖 $W_t = M^*$ 的情况（此时上述表达式会导致除以 0），我们正式将漂移函数定义为。

$$D_t := \max\left\{(W_t - M^*)^{19} \cdot 2\Delta_t, \frac{(W_t - M^*)^{21}}{2\Delta_t}\right\}. \tag{2}$$

为便于阅读，下面我们使用（1），但应理解为形式上指的是（2）。

我们将证明，在第 4 行 for 循环的每次迭代中，D_t 都会（在期望意义下）按一个常数因子递减。注意，当 D_t 达到 0 时，我们便有 $W_t = M^*$，因此问题得以解决。

▶ **引理 43**（负漂移）。对于任意 $d > 0$ 和 $t \geq 0$，都有上述结论。

$$\mathbf{E}(D_{t+1} \mid D_t = d) \leq 0.7 \cdot d.$$

在证明引理 43 之前，我们先看看如何由它推出引理 42。为此，我们将使用以下工具：

▶ **定理 44**（乘法漂移，参见例如 [41, 定理 18]）。令 $(X_t)_{t\geq 0}$ 为一列非负随机变量，其状态空间 S 为有限的非负整数集合。假设 $X_0 = s_0$，并且存在 $\delta > 0$ 使得对于所有 $s \in S \setminus \{0\}$ 和所有 $t \geq 0$，$\mathbf{E}(X_{t+1} \mid X_t = s) \leq (1-\delta)s$。那么，对于所有 $r \geq 0$，

$$\mathbf{P}(X_r > 0) \leq e^{-\delta \cdot r} \cdot s_0.$$

证明。 根据马尔可夫不等式，$\mathbf{P}(X_r > 0) = \mathbf{P}(X_r \geq 1) \leq \mathbf{E}(X_r)$。应用界$\mathbf{E}(X_{t+1} \mid X_t = d) \leq (1-\delta)d$ 共 r 次，我们得到

$$\mathbf{P}(X_r > 0) \leq (1-\delta)^r \cdot s_0 \leq \exp(-\delta r) \cdot s_0.$$ ◀

引理 42的证明。伪代码见算法 7。首先分析运行时间。在for 循环的每次迭代中，要么满足 $W_t \le 24$ 且我们直接使用至多 24 次 DetectNegCycle 调用来解决问题，要么调用一次 TestScale。根据引理40，每次 TestScale 调用所需时间为 $O(T_{\mathrm{RSSSP}}(m,n))$；而我们只调用一次 DetectNegCycle，其总运行时间根据推论34 为 $O(T_{\mathrm{RSSSP}}(m,n)\log\ n)$。由于 $T = \Theta(\log(nW))$，总体运行时间上界为 $O(T_{\mathrm{RSSSP}}(m,n)\ \log\ n + T_{\mathrm{RSSSP}}(m,n)\ \log(nW))$，如所述。

现在我们分析正确性。注意，在每次迭代中， G_t 都等价于 G ，因为我们修改图的唯一方式是添加势函数（见引理 16）。因此，如果某个时刻我们有 $W_t \le 24$ ，那么正确答案就在范围 $0 \le M^* \le 24$内。第 7 行的 for 循环会通过调用 DetectNegCycle，穷举检查哪个值是正确的。根据推论 34，这以高概率正确。

现在假设算法未在第8行终止。我们声称，最终漂移D_T 以高概率为零。注意，这意味着正确性，因为 $D_T = 0$ 当且仅当 $W_T = M^*$ （要看出这一点，请注意 $\Delta_T \ge 1$ 这是由于第12行）。为了证明这一点，我们将使用定理44。注意，引理 43告诉我们$\mathbf{E}(D_{t+1} \mid D_t = d) \le 0.7d$。此外，我们可以将初始漂移 D_0 界定为

$$D_0 = (W - M^*)^{20} \cdot \max\left\{\frac{2\Delta_0}{W - M^*}, \frac{W - M^*}{2\Delta_0}\right\} \le (W - M^*)^{21} \cdot 2\Delta_0 \le 4W^{21}.$$

因此，由定理 44 可得$\mathbf{P}(D_T > 0) \le \exp(-0.7T) \cdot 4W^{21}$。由于 $T = \Theta(\log(nW))$，我们得出$\mathbf{P}(D_T > 0) \le n^{-\Omega(1)}$，至此证明完成。◀

引理 43 的证明。聚焦于第 t 行 for 循环中的迭代。令 E_1 表示我们得到一个势函数 ϕ 的事件（即第 9 行的 if 语句成立），并令 $E_2 := \neg E_1$表示其补事件。我们先观察参数 W_{t+1} 和 Δ_{t+1} 如何随 E_1 或 E_2 是否发生而变化。

▷ 断言 45. 若发生 E_1 ，则有 $W_{t+1} \le W_t - \frac{\Delta_t}{4}$, 且 $\Delta_{t+1} = 2\Delta_t$。

证明。若第 9 行对 TestScale 的调用返回势函数 ϕ，则我们设置 $G_{t+1} = (G_t)_\phi$和 $\Delta_{t+1} = 2\Delta_t$。注意到 $G_t^{+W_t-\Delta_t}$的最小权重为 Δ_t。因此，引理 40 保证 $(G_t)_\phi^{+W_t-\Delta_t}$的最小权重至少为 $-\frac{3}{4}\Delta_t$。由于 $G_{t+1} = (G_t)_\phi$是通过从 $(G_t)_\phi^{+W_t-\Delta_t}$中的每条边的权重减去 $W_t - \Delta_t$ 来定义的，我们得到 $-W_{t+1} \ge -W_t + \frac{1}{4}\Delta_t$。◁

▷ 断言 46. 若出现 E_2 ，则 $W_{t+1} = W_t$ 和 $\Delta_{t+1} = \max\{1, \Delta_t/2\}$ 以及 $D_{t+1} \le 2D_t$。

证明。前两个陈述可由第12行直接得出。为证明第三个陈述，对于函数 $f(x) := \max\{x, 1/x\}$，我们注意到，如果 x 和 $y > 0$ 相差至多 2 倍，那么 $f(x)$ $f(y)$ 也相差至多 2 倍。现在利用 $D_t = (W_t - M^*)^{20} \cdot f(2\Delta_t/(W_t - M^*))$。由于 $\Delta_t \ge 1$， Δ_t 和 Δ_{t+1} 相差至多 2 倍，因此 D_t 和 D_{t+1} 也相差至多 2 倍。◁

有了这些断言，我们接下来界定 D_{t+1} 在 $D_t > 0$ 时的漂移。回想一下，我们定义了

$$D_t = (W_t - M^*)^{20} \cdot \max\left\{\frac{2\Delta_t}{W_t - M^*}, \frac{W_t - M^*}{2\Delta_t}\right\}. \tag{1}$$

注意，始终有 $W_t \ge M^*$和 $W_{t+1} \ge M^*$。此外，由于 $D_t > 0$，我们可以假设 $W_t - M^* > 0$，因为否则 $W_t - M^* = 0$ ，从而 $D_t = 0$。我们根据在（1）中取得最大值的项进行分类讨论。

情形 1 $\Delta_t \geq \frac{1}{2}(W_t - M^*)$：那么，有 $D_t = (W_t - M^*)^{19} \cdot 2\Delta_t$。如果 E_1 发生，那么根据断言 45，有 $\Delta_{t+1} \geq \Delta_t \geq \frac{1}{2}(W_t - M^*) \geq \frac{1}{2}(W_{t+1} - M^*)$。因此，利用（1），我们可以将漂移界定为 D_{t+1}：

$$\begin{aligned} D_{t+1} &= (W_{t+1} - M^*)^{19} \cdot 2\Delta_{t+1} \\ &\leq (W_t - M^* - \tfrac{\Delta_t}{4})^{19} \cdot 4\Delta_t \\ &\leq (W_t - M^* - \tfrac{1}{8}(W_t - M^*))^{19} \cdot 4\Delta_t \\ &\leq (\tfrac{7}{8})^{19} \cdot 2D_t \leq 0.16 D_t, \end{aligned}$$

其中，第一个不等式用到了断言 45；第二个不等式则由情况 1 的假设得出，即我们有 $\frac{\Delta_t}{4} \geq \frac{1}{8}(W_t - M^*)$。

如果出现 E_2，则我们进一步进行分类讨论：

情况 1。1 $\Delta_t > W_t - M^*$：注意，如果 $\Delta_t = 1$，那么由于 W_t 和 M^* 是整数，因此可得 $W_t = M^*$，从而 $D_t = 0$，这与 $D_t > 0$ 的假设相矛盾。因此，我们可以假设 $\Delta_t \geq 2$。特别地，根据断言 46，我们有 $\Delta_{t+1} = \frac{1}{2}\Delta_t > \frac{1}{2}(W_t - M^*) = \frac{1}{2}(W_{t+1} - M^*)$。因此，根据（1），我们可以将漂移 D_{t+1} 表示为

$$D_{t+1} = (W_{t+1} - M^*)^{19} \cdot 2\Delta_{t+1} = (W_t - M^*)^{19} \cdot \Delta_t = \frac{D_t}{2}.$$

情况 1。2 $\Delta_t \leq W_t - M^*$：注意，在这种情况下，$G^{+W_t - \Delta_t}$ 不包含负环。此外，我们可以假设 $W_t > 24$，因为否则问题可直接在第 7 行中得到解决。因此，根据引理 40，我们有 $\mathbf{P}(E_2) \leq 0.01$。此外，根据断言 46，我们有 $D_{t+1} \leq 2D_t$。

结合上述内容，我们得出，对于情况 1，有

$$\begin{aligned} \mathbf{E}(D_{t+1} \mid D_t) &\leq \mathbf{P}(E_1)\,\mathbf{E}(D_{t+1} \mid D_t, E_1) + \mathbf{P}(E_2)\,\mathbf{E}(D_{t+1} \mid D_t, E_2) \\ &\leq 1 \cdot 0.16 D_t + \max\left\{1 \cdot \tfrac{1}{2}D_t, 0.01 \cdot 2D_t\right\} \leq 0.66 D_t. \end{aligned}$$

情况 2 $\Delta_t < \frac{1}{2}(W_t - M^*)$：那么，有 $D_t = (W_t - M^*)^{21}/(2\Delta_t)$。若 E_2 出现，则根据情况 1.2 中的相同论证可知 $D_{t+1} \leq 2D_t$ 且 $\mathbf{P}(E_2) \leq 0.01$。

如果改为出现 E_1，那么我们进一步进行分类讨论：

情况 2。1 $\Delta_{t+1} < \frac{1}{2}(W_{t+1} - M^*)$：然后利用 (1)，可得

$$D_{t+1} = \frac{(W_{t+1} - M^*)^{21}}{2\Delta_{t+1}} \leq \frac{(W_t - M^*)^{21}}{4\Delta_t} = \frac{D_t}{2},$$

where the inequality holds due to Claim 45.

情况 2。2 $\Delta_{t+1} \geq \frac{1}{2}(W_{t+1} - M^*)$：那么有 $D_{t+1} = (W_{t+1} - M^*)^{19} \cdot 2\Delta_{t+1}$。由于根据情况 2 的假设，我们有 $(W_t - M^*)/(2\Delta_t) \geq 1$，并且根据断言 45，我们有 $\Delta_{t+1} = 2\Delta_t$，因此可以将 D_{t+1} 界定为

$$\begin{aligned} D_{t+1} &= (W_{t+1} - M^*)^{19} \cdot 2\Delta_{t+1} \\ &\leq (W_{t+1} - M^*)^{19} \cdot 4\Delta_t \cdot \left(\frac{W_t - M^*}{2\Delta_t}\right)^2 \\ &= (W_{t+1} - M^*)^{19} \cdot (W_t - M^*)^2 \cdot \tfrac{1}{\Delta_t}. \end{aligned} \tag{3}$$

根据断言 45，我们有 $W_{t+1} \le W_t - \frac{\Delta_t}{4}$。因此，可以将 $W_{t+1} - M^*$ 界定为

$$\begin{aligned} W_{t+1} - M^* &= \tfrac{16}{17}(W_{t+1} - M^*) + \tfrac{1}{17}(W_{t+1} - M^*) \\ &\le \tfrac{16}{17}(W_t - M^* - \tfrac{\Delta_t}{4}) + \tfrac{1}{17}(W_{t+1} - M^*) \\ &= \tfrac{16}{17}(W_t - M^*) - \tfrac{16}{17} \cdot \tfrac{\Delta_t}{4} + \tfrac{1}{17}(W_{t+1} - M^*). \end{aligned} \tag{4}$$

根据断言 45 和情况 2.2 的假设，我们有 $2\Delta_t = \Delta_{t+1} \ge \frac{1}{2}(W_{t+1} - M^*)$。这意味着 $\frac{\Delta_t}{4} \ge \frac{1}{16}(W_{t+1} - M^*)$。将其代入 (4)，得到

$$\begin{aligned} W_{t+1} - M^* &\le \tfrac{16}{17}(W_t - M^*) - \tfrac{1}{17}(W_{t+1} - M^*) + \tfrac{1}{17}(W_{t+1} - M^*) \\ &= \tfrac{16}{17}(W_t - M^*). \end{aligned} \tag{5}$$

最后，结合 (3) 和 (5)，得到

$$\begin{aligned} D_{t+1} &\le (W_{t+1} - M^*)^{19}(W_t - M^*)^2 \cdot \tfrac{1}{\Delta_t} \\ &\le (\tfrac{16}{17})^{19}(W_t - M^*)^{21} \cdot \tfrac{1}{\Delta_t} \\ &= (\tfrac{16}{17})^{19} \cdot 2 \cdot D_t \\ &\le 0.65 D_t \end{aligned}$$

结合所讨论的各个子情况，我们得出情况 2 满足

$$\begin{aligned} \mathbf{E}(D_{t+1} \mid D_t) &\le \mathbf{P}(E_1)\,\mathbf{E}(D_{t+1} \mid D_t, E_1) + \mathbf{P}(E_2)\,\mathbf{E}(D_{t+1} \mid D_t, E_2) \\ &\le 1 \cdot \max\left\{\tfrac{1}{2}D_t, 0.65 D_t\right\} + 0.01 \cdot 2D_t \le 0.67 \cdot D_t. \end{aligned}$$

由于情况 1 和情况 2 已涵盖所有情况，证明得证。 ◀

5.4 汇总所有内容

现在我们将各个部分整合起来，证明我们的主要定理。

▶**定理 1**（负权单源最短路径）。 存在一种拉斯维加斯算法，给定一个有向图 G 和一个源点 s，要么从 s 计算出一棵最短路径树，要么在 G中找到一个负环，其运行时间为 $O((m + n \log\log n) \log^2 n \log(nW))$，且以高概率（以及在期望意义下）成立。

证明。 该算法交替执行以下两个步骤，并在每个步骤超过 $O((m + n \log\log n) \log^2 n \log(nW))$的时间预算后中断该步骤：

1. 运行 单源最短路径(G, s)。如果该算法在规定时间内完成并返回一棵最短路径树，我们会检查这棵最短路径树是否正确（通过松弛所有边并测试树中的任何距离是否发生变化）；如果正确，则返回这棵最短路径树。否则，继续执行步骤 2。

2. 运行FindNegCycle(G) （使用引理 42 来实现 阈值）。如果该算法在规定时间内完成并返回一个负环，我们会验证输出确实是一个负环；如果是，则返回这个负环。否则，继续执行步骤 1。

该算法显然是正确的：每当它终止时，都会报告正确的解。我们来关注运行时间。我们区分两种情况：首先，假设 G 不包含负环。根据定理 29，步骤 1 以高概率在 $O((m + n \log\log n) \log^2 n \log(nW))$ 时间内运行，并且在这种情况下不会被中断。此外，单源最短路径算法返回一个

以高概率得到正确的最短路径树，从而在步骤 1 仅迭代一次后终止算法。

另一方面，假设 G 包含一个负环。算法会执行步骤 1，而在这种情况下这只是徒劳的工作，但仅需耗费 $O((m+n\log\log n)\log^2 n\log(nW))$ 的时间。随后，根据引理 36 和引理 42，步骤 2 的一次执行以高概率在时间预算内完成。此外，由于引理 36 是一个拉斯维加斯算法，它会返回一个真正的负环，算法随即终止。

前两段证明了，依次运行步骤 1 和步骤 2 后，算法以高概率在 $O((m+n\log\log n)\log^2 n\log(nW))$ 的时间内成功终止。由于我们会独立地重复这些步骤，直到算法终止，因此同样的界也适用于预期运行时间。◀

接下来，我们将前面的定理 1 作为黑盒来证明定理 2。

▶ **定理 2**（负权单源距离）。存在一种拉斯维加斯算法，给定一个有向图 G 和一个源点 $s\in V(G)$，可计算从 s 到图中所有其他顶点的距离（这些距离可能为 $-\infty$ 或 ∞），运行时间为 $O((m+n\log\log n)\log^2 n\log(nW))$，且该时间界以高概率成立（期望时间也满足该界）。

证明。首先，从图中移除所有从 s 不可达的顶点，并为每个此类顶点返回距离 ∞。然后在 G 中用 $O(m+n)$ 时间计算强连通分量集合 $C_1,\dots,C_\ell$。对于每个强连通分量 C_i，运行定理 1 中的单源最短路径算法，检查 $G[C_i]$是否包含负环。对于包含在含有负环的强连通分量中的每个顶点，我们返回距离 $-\infty$（由于该强连通分量可从 s 到达且包含负环，因此我们可以无限循环）。类似地，对于所有可从某个 $-\infty$-距离顶点到达的顶点，报告 $-\infty$。移除所有距离为 $-\infty$的顶点后，剩余图中不再包含负环。因此，我们可以在剩余图上运行单源最短路径算法，计算尚未求得的距离。

令 n_i 和 m_i 分别表示子图 $G[C_i]$ 中顶点和边的数量。则总运行时间为

$$\begin{aligned}
&O\left(T_{\mathrm{SSSP}}(m,n,W)+\sum_i T_{\mathrm{SSSP}}(m_i,n_i,W)\right)\\
&\quad=O\left(\left(m+n\log\log n+\sum_i m_i+\sum_i n_i\log\log n\right)\log n^2\log(nW)\right)\\
&\quad=O((m+n\log\log n)\log^2 n\log(nW)),
\end{aligned}$$

使用 $\sum_i m_i\le m$ 和 $\sum_i n_i\le n$。◀

6 最小环均值

在本节中，我们证明定理3，即我们给出用于计算图 G的最小环均值的 $O((m+n\log\log n)\log^2(n)\log(nW))$- 时间算法。

给定有向图 G，我们将 $\mu^*(G)$记为最小环均值的值，即 $\mu^*(G):=\min_C \bar{w}(C)$。为了设计我们的算法，以下最小环均值刻画将很有用：

▶**引理 47.** 设 G 为一个有向图。那么，

$$\mu^*(G)=-\min\{Q\in\mathbf{Q}\mid G^{+Q}\ \textit{contains no negative cycle}\}.$$

证明。根据定义，我们有 $\mu^*(G) = \min_C \bar{w}(C)$。等价地， $\mu^*(G)$ 是满足以下条件的最大有理数 μ ：对 G 中的所有环 C ， $\mu \leq w(C)/|C|$ 均成立。特别地，对所有环 C，$w(C) - \mu \cdot |C| \geq 0$ 均成立，这等价于 $G^{-\mu}$ 不含负环。◀

回顾一下，Threshold(G) 计算满足 G^{+M^*} 不含负环的最小整数 $M^* \geq 0$ （定义 35）。这与引理 47 给出的最小环均值刻画非常相似，不同之处在于，后者是在不一定非负的有理数上取最小值。为解决这一问题，我们将使用以下简单命题：

▶ **命题 48.** 设 G 为有向图， $a \geq 1$、 $b \geq 0$ 为整数。设 H 为 G 的一个副本，其中每条边的权重为 $w_H(e) := a \cdot w_G(e) + b$。设 C 是 G 中任意一个环。那么，$\bar{w}_H(C) = a \cdot \bar{w}_G(G) + b$。

证明。注意， C 在 H 中的权重恰好为 $w_H(C) = a \cdot w_G(C) + b \cdot |C|$。因此， C 在 H 中的环均值等于 $\bar{w}_H(C) = a \cdot w_G(C)/|C| + b = a \cdot \bar{w}_G(C) + b$。◀

▶**命题 49.** 设 C 和 C' 是有向图 G 中的两个环，且它们的均值不同，即 $\bar{w}(C) \neq \bar{w}(C')$。那么， $|\bar{w}(C) - \bar{w}(C')| \geq 1/n^2$。

证明。 根据定义，我们可以将 $|\bar{w}(C) - \bar{w}(C')|$ 表示为

$$\left|\frac{w(C)}{|C|} - \frac{w(C')}{|C'|}\right| = \left|\frac{w(C)|C'| - w(C')|C|}{|C| \cdot |C'|}\right| \geq \frac{1}{|C||C'|},$$

这里我们用到了 $\bar{w}(C) \neq \bar{w}(C')$。由于 $|C|, |C'| \leq n$，因此该值至少为 $1/n^2$。 ◀

我们将使用下面的引理，它是引理 42 的一种拉斯维加斯实现。

▶**引理 50.** 设 G 是一个有向图。存在一种拉斯维加斯算法，可以计算 *Threshold*(G) （参见定义 *35*），并以高概率（且在期望意义下）在时间 $O((m + n \log\log n) \log^2 n \log(nW))$内运行。

证明。 该算法使用引理 42 计算 $M^* = \text{阈值}(G)$ 。根据定义，这将返回最小整数 M^* 使得 G^{+M^*} 以高概率不包含负环（回顾定义 35）。要将其转化为拉斯维加斯算法，我们需要验证输出是否正确。为此，我们添加一个源顶点 s ，并将其通过权重为 0 的边连接到所有其他顶点，然后使用定理 1 测试 G^{+M^*} 是否不包含负环，以及 G^{+M^*-1} 是否包含负环。如果任一测试失败，算法将重新启动。

该过程的正确性源于定理 1 是一个拉斯维加斯算法。对于运行时间，注意到，对引理 42 的调用（使用定理 18 的 $T_{\text{RSSSP}}(m, n)$ 界）以及对定理 1的调用，其运行时间为 $O((m + n \log\log n) \log^2 n \log(nW))$。此外，引理42以高概率保证值 M^* 是正确的。因此，该算法以高概率在 $O((m + n \log\log n) \log^2 n \log(nW))$-时间内终止。◀

▶**定理 3** (最小环均值)。 存在一种拉斯维加斯算法，给定有向图 G ，可找到平均权重最小的环 C $\bar{w}(C) = \min_{C'} \bar{w}(C')$，运行时间为 $O((m + n \log\log n) \log^2 n \log(nW))$ ，且以高概率（并且在期望意义下）完成。

算法8给定一个图 G 该过程以高概率返回平均权重最小的环 C 。参见定理 3。

---

1 procedure最小环均值(G)**2** 令 L 为 G中的最大权重**3** 令 H 为 G 的一个副本，其边权为 $w_H(e) \leftarrow n^2 \cdot w_G(e) - n^3 L$。**4** 使用引理 50 计算 $M^* \leftarrow$ 阈值(H) **5** $C \leftarrow$查找负环(H^{+M^*-1})**6** 返回 C

证明。 我们通过将 H 中的每条边的权重修改为 G 来构造图 $n^2 w(e) - n^3 L$，其中 L是 G中的最大边权。然后，我们计算 $M^* := $ 阈值(H) 使用引理 50。最后，我们使用引理 36 在 H^{+M^*-1} 中找到一个负环。伪代码见算法 8。

运行时间主要由对阈值和 FindNegCycle 的调用决定。根据引理 50，调用阈值所需的时间为 $O((m + n \log\log n) \log^2 n \log(nW))$，且以高概率成立。根据引理 36，调用 FindNegCycle（使用引理 42 实现阈值，并使用定理 18 对 $T_{\mathrm{RSSSP}}(m, n)$ 进行界定）所需的时间为 $O((m + n \log\log n) \log^2 n \log(nW))$，且同样以高概率成立。因此，该算法的运行时间符合所声称的界。

为分析正确性，注意命题48 蕴含：一个环 C 是 $\bar{w}_G(C)$ 的最小化者，当且仅当它是 $\bar{w}_H(C)$ 的最小化者。因此，只需在 H 中找到均值最小的环。我们将论证，该算法找到的环就是最小化者。

▷断言 51. 第 4 行计算出的值 M^* 满足 $M^* = \lceil -\mu^*(H) \rceil$。

证明。 我们注意到 H 中的最小环均值是非正的，即 $\mu^*(H) \leq 0$。为说明这一点，注意任意环 C 在 G 中的权重至多为 $w_G(C) \leq nL$。因此，根据我们在 H中设置权重的方式， H 中的任意环的权重为 $w_H(C) = n^2 w_G(C) - n^3 L|C| \leq n^3 L - n^3 L = 0$。这意味着在引理 47 中，我们可以对 $Q \geq 0$ 进行最小化，即

$$\mu^*(H) = -\min\{0 \leq Q \in \mathbf{Q} \mid H^{+Q} \text{ contains no negative cycle}\}. \tag{6}$$

回顾一下，根据阈值(H)的定义， M^* 是使 H^{+M^*} 没有负环的最小非负整数，即

$$M^* = \min\{0 \leq M \in \mathbf{Z} \mid H^{+M} \text{ contains no negative cycle}\}. \tag{7}$$

结合 (6) 和 (7)，我们得出 $M^* = \lceil -\mu^*(H) \rceil$，如所述。 ◁

因此， H^{+M^*-1} 确实包含一个负环。根据引理 50，对阈值的调用是正确的。因此，H^{+M^*-1} 包含一个负环，并且根据引理 36，对 FindNegCycle 的调用是正确的。设 C 为第 5 行得到的环。由于它在 H^{+M^*-1}中的权重为负，因此它在 H 中的权重小于 $-|C|(M^* - 1)$。因此，有 $\bar{w}_H(C) < -M^* + 1$。此外，由于 H^{+M^*}不包含负环，每个环 C'的平均权重都满足 $\bar{w}_H(C') \geq -M^*$。

现在考虑一个最小平均环 C'。正如我们所见，我们有

$$-M^* \leq \bar{w}_H(C') \leq \bar{w}_H(C) < -M^* + 1. \tag{8}$$

假设为导出矛盾， $\bar{w}_H(C) \neq \bar{w}_H(C')$。那么根据命题 49，我们有 $|\bar{w}_G(C) - \bar{w}_G(C')| \geq 1/n^2$，并且根据命题 48，有 $\bar{w}_H(C) = n^2 \cdot w_G(C) - n^3 L$

且 $\bar{w}_H(C') = n^2 \cdot w_G(C') - n^3 L$。结合这些事实，我们得到 $|\bar{w}_H(C) - \bar{w}_H(C')| \geq 1$。这与方程（8）矛盾。因此，我们得到 $\bar{w}_H(C) = \bar{w}_H(C')$，所以所计算的环 C 是 $\bar{w}_H(C)$ 的最小化解，因而也是 $\bar{w}_G(C)$ 的最小化解。◀

7 低直径分解

本节我们建立强低直径分解（低直径分解）。回顾一下，在强低直径分解中（如定义 4 所定义），目标是选取一个小的边集 S，使得移除 S 中的边后，剩余图中的每个强连通分量的直径都有界。我们的结果如下定理所示：该定理证明了强低直径分解的存在性（这一点已由 [11] 知晓），并且可以高效计算（这一点此前尚未解决）：

▶ **定理 5**（强低直径分解）。存在一种开销为 $O(\log^3 n)$ 的强低直径分解，可在 $O((m + n \log\log n) \log^2 n)$ 时间内计算，并以高概率（以及在期望意义下）成立。

7.1 重顶点和轻顶点

在该算法中，我们将根据这些顶点的出球和入球大小，区分重型 和*Light*顶点。为了将顶点分类为重型或Light，我们依赖以下简单的引理：

▶ **引理 52**（估计球的大小）。设 $\varepsilon > 0$。给定一个具有非负边权的有向图 G 和 $r > 0$，我们可以对每个顶点 v 近似计算 $|B^{out}(v, r)|$，加性误差为 εn。该算法以高概率成功，运行时间为 $O(\varepsilon^{-2} \log n \cdot (m + n \log\log n))$。

证明。随机采样顶点 $v_1, \ldots, v_k \in V(G)$（允许重复），其中 $k := 5\varepsilon^{-2} \log n$。对所有 $i \in [k]$ 计算 $B^{in}(v_i, r)$。使用 Dijkstra 算法和 Thorup 优先队列 [23, 57],，此步骤的运行时间为 $O(k \cdot (m + n \log\log n)) = O(\varepsilon^{-2} \log n \cdot (m + n \log\log n))$。现在，对每个顶点 v，返回其估计值

$$b(v) := \frac{n}{k} \cdot |\{\, i \in [k] : v \in B^{in}(v_i, r) \,\}|.$$

我们断言该估计值是准确的。令 I_i 表示$v_i \in B^{out}(v, r)$ 是否成立的指示变量，并令 $I := \sum_{i=1}^{k} I_i$。那么，随机变量 $b(v)$ 恰好为

$$b(v) = \frac{n}{k} \cdot I.$$

注意，$\mathbf{P}(I_i = 1) = |B^{out}(v, r)|/n$。因此，期望为

$$\mathbf{E}(b(v)) = \frac{n}{k} \cdot \sum_{i=1}^{k} \mathbf{P}(I_i = 1) = \frac{n}{k} \cdot \frac{k}{n} \cdot |B^{out}(v, r)| = |B^{out}(v, r)|.$$

利用 Chernoff 界可得 $\mathbf{P}(|I - \mathbf{E}(I)| > a) < 2\exp(-2a^2/k)$。对于 $a := \varepsilon k$，我们得到

$$\mathbf{P}(|b(v) - \mathbf{E}(b(v))| > \varepsilon n) = \mathbf{P}(|I - \mathbf{E}(I)| > \varepsilon k) < 2\exp(-2\varepsilon^2 k) \leq 2n^{-10}.$$

因此，以高概率计算出的估计是准确的。◀

▶ **引理 53** (重/轻分类)。存在一种算法 $\textsc{Light}(G, r)$，给定一个有向图 G 和一个半径 r，返回一个集合 $L \subseteq V(G)$，满足以下性质：对于所有 $v \in L$，都有 $|B^{out}(v, r)| \leq \frac{7}{8} n$。

- 对所有 $v \in V(G) \setminus L$，均有 $|B^{out}(v,r)| \geq \frac{3}{4}n$。 –
- *Light*(G,r) 的运行时间为 $O((m+n\log\log n)\log n)$。

证明。我们使用参数 $\varepsilon := \frac{1}{16}$ 运行前面的引理 52，并令 L 为估算球大小不超过 $\frac{13}{16}n$ 的顶点子集。以高概率，这些估算的加性误差至多为 $\varepsilon n = \frac{1}{16}n$。因此，任意顶点 $v \in L$ 都满足 $|B^{out}(v,r)| \leq \frac{13}{16}n + \frac{1}{16}n = \frac{7}{8}n$，且任意顶点 $v \in V(G) \setminus L$ 都满足 $|B^{out}(v,r)| \geq \frac{13}{16}n - \frac{1}{16}n = \frac{3}{4}n$。运行时间由引理 52 主导，其运行时间为 $O((m+n\log\log n)\log n)$，如所述。◀

7.2 强低直径分解

强低直径分解的工作方式如下：令 $R = {}_{10}{}^{D}{}_{\log n_0}$。首先，我们在 G 上使用半径为 R 的引理 53，计算集合 L^{out}；然后在反向图上使用半径为 R 的引理 53，计算集合 L^{in}。我们将 L^{out} 中的顶点称为出轻，将 L^{in} 中的顶点称为入轻，并将 $V(G)\setminus(L^{out}\cup L^{in})$ 中的顶点称为重。接着，我们区分两种情况：

- 重情形：如果存在重顶点 $v \in V(G)\setminus(L^{out}\cup L^{in})$，我们计算顶点集 W，其中的顶点既能在距离 R内到达 v，也能在距离 R内从 v 到达，即$W = B^{out}(v,R) \cap B^{in}(v,R)$。令 Tout T^{in}分别表示从 v到 W以及从 W 到 v的最短路径树。令 C 为 T^{out} 和 T^{in}中的顶点并集。我们收缩 C（也就是说，将 C 中的所有顶点替换为一个超顶点），并考虑剩余的（多重）图 G/C。我们递归地计算 G/C中的强低直径分解，得到一个边集 S。在 S 中，我们展开所有与超顶点相连的边（即，对于任意一条边 $(v,u) \in E(G)$，如果它在收缩后的图中变成了一条边 (C,u)，我们就将 (C,u) 恢复为 (v,u)），并返回 S。
- 轻情形：如果不存在重顶点，那么每个顶点都是出轻或入轻。对于每个顶点 v（比如说，它是出轻的），因此我们可以按标准方式进行：采样一个半径 r，该半径取自参数为 $O(\log n/D)$ 的几何分布，切断离开 B^{out} 的边（v，r），并在球 B^{out}（v，r）的内部和外部上递归。

我们在算法 9 中总结了带有精确参数的伪代码。在本节中，我们用 n_0 表示原始图的大小，用 n 表示当前图 G 的大小（在算法当前的递归调用中）。

▶ **引理 54** (算法 9 的强直径)。 以高概率，*StrongLDD*(G,D)要么返回失败，要么返回边集 $S \subseteq E(G)$，使得 $G \setminus S$ 的每个强连通分量 C 的直径至多为 D，即 $\max_{u,v\in C} \mathrm{dist}^{G[C]}(u,v) \leq D$。

证明。 以高概率，重轻分类在 StrongLDD(G,D) （以及所有递归调用）中均能正确运行。我们以该事件为条件，并将分类视为完美分类。

和之前一样，我们需要区分重情形和轻情形。在重情形中，令 v 为重顶点，并令 W，Tout T^{in}, C如算法中所述。我们声称，诱导子图 $G[C]$ 的直径至多为 $4R$。任取一个顶点 $x \in C$；只需证明 dist$^{G[C]}(v,x) \leq 2R$ 且 dist$^{G[C]}(x,v) \leq 2R$。我们证明前一个结论，后一个结论从略。有两种简单情形：要么有 $x \in T^{out}$，此时我们立即得到 dist$^{G[C]}(v,x) \leq R$（因为 Tout中的任意路径长度至多为 R）；要么有 $x \in T^{in}$，此时存在某个中间顶点 $y \in W$，使得 dist$^{G[C]}(y,x) \leq R$。但此时也有 dist$^{G[C]}(v,y) \leq R$，结合起来便得到 dist$^{G[C]}(v,x) \leq 2R$，如所声称。

算法 9 强低直径分解（见定理 5）的实现：要么返回边集 $S \subseteq E(G)$，要么失败。

过程 StrongLDD(G, D)
  如果 $n \le 100$，**则返回** $E(G)$
  令 $R := \frac{D}{10 \log n}$
  计算 $L^{out} \leftarrow$ Light(G, R)
  计算 $L^{in} \leftarrow$ Light(G^{rev}, R)（此处，G^{rev} 是具有反向边的图 G）
  （重情形）
  如果 存在重顶点 $v \in V(G) \setminus (L^{out} \cup L^{in})$ **则**
    令 $W \leftarrow B^{out}(v, R) \cap B^{in}(v, R)$
    计算最短路径树 T^{out}，从 v 到 W，以及 T^{in}，从 W 到 v
    令 C 为 T^{out} 中的顶点并集 $, T^{in}$
    $S \leftarrow$ StrongLDD$(G/C, D - 4R)$
    返回 S 在展开所有边之后
  （轻情形）
  $S \leftarrow \emptyset$
  当存在 $v \in V(G) \cap L^{out}$ **时**
    $r \sim$ 几何分布$(R^{-1} \cdot 10 \log n_0)$
    如果 $r > R$，**则返回** 失败
    $S \leftarrow S \cup \partial B^{out}(v, r) \cup$ StrongLDD$(G[B^{out}(v, r)], D)$
    $G \leftarrow G \setminus B^{out}(v, r)$
  当存在 $v \in V(G) \cap L^{in}$ **时，执行**
    $r \sim$ 几何分布$(R^{-1} \cdot 10 \log n_0)$
    如果 $r > R$，**则返回** 失败
    $S \leftarrow S \cup \partial B^{in}(v, r) \cup$ StrongLDD$(G[B^{in}(v, r)], D)$
    $G \leftarrow G \setminus B^{in}(v, r)$
  return S

回想一下，该算法将 C 中的顶点收缩起来，并在剩余的多重图上以参数 $D - 4R$ 计算强低直径分解 S。根据归纳法，我们假设递归调用为 G/C 计算出了正确的强分解。为了说明该分解对于 G 也正确，任取同一强连通分量中的两个顶点 u，v，它们位于 $G \setminus S$ 中。我们有 $\text{dist}^{(G/C)\setminus S}(u, v) \le D - 4R$。如果 G/C 中的最短 u-v 路径没有经过超顶点，那么立即有 $\text{dist}^{G\setminus S}(u, v) \le D - 4R \le D$。如果最短路径经过超顶点，那么我们可以用一条长度为 $\text{diam}(G[C]) \le 4R$ 的路径替换经过 C 的那段路径。因此 $\text{dist}^{G\setminus S}(u, v) \le D - 4R + 4R \le D$。

轻情形的正确性与 [11]，已知的低直径分解完全相同，也与引理 8 类似：对于算法划出的每个球 $B^{out}(v, r)$（或 $B^{in}(v, r)$），我们会移除所有出边 $\partial B^{out}(v, r)$（或分别移除所有入边 $\partial B^{in}(v, r)$）。因此，剩余图中的两个顶点 x, y 只有在二者都属于 $x, y \in B^{out}(v, r)$ 或二者都属于 $x, y \notin B^{out}(v, r)$ 时，才属于同一个强连通分量。算法继续对 $B^{out}(v, r)$ 之外的所有顶点执行循环，并在 $B^{out}(v, r)$ 内递归。根据归纳法，两次调用都会成功，并将直径缩小至至多 D。

最终，算法会到达一种基本情况：G 中仅包含常数数量的节点和边——在这种情况下，我们可以选择将 S 设为整个边集。◀

▶ **引理 55** (算法 9 的稀疏命中)。对于任意边 $e \in E(G)$，e 的概率

包含在 *StrongLDD*（G、D）的输出中的权重至多为 $O(^{w(e)D\cdot\log^3(n_0)}+{}^{1}\mathrm{poly}(n))$。

证明。 在此证明中，我们以初始计算得到的重/轻分类正确这一事件为条件。由于该事件以高概率发生，因此我们只会将所有边的命中概率增加 $^{1}\mathrm{poly}(n)$。

设 $p(n, w, D)$ 是权重为 w 的边包含在 StrongLDD(G, D) 输出中的概率上界，其中 G 是一个 n 顶点图。我们通过归纳证明 $p(n, w, D) \le \frac{w}{D}\cdot 1000\log(n_0)\log^2(n)$，这正是所需结论。我们在算法 9 中区分重情形和轻情形。

轻情形。 假设算法进入轻情形（也就是说，没有顶点被分类为重顶点）。考虑某条权重为 $e=(x, y)$ 的边 $w=w(e)$。对于每次迭代，我们区分三种情况。假设当前迭代选取了一个出轻顶点 v。

- x、$y \notin B^{out}(v,r)$：边 e 在本次迭代中未被触及，并仍是图 G 的一部分。它是否包含在输出中取决于后续迭代。$x \in B^{out}(v,r)$ 和 $y \notin B^{out}(v,r)$：在这种情况下，
- $e \in \partial B^{out}(v,r)$，因此该边被包含在 S 中。$y \in B^{out}(v,r)$：该边未被包含在 $\partial B^{out}(v,r)$ 中。不过，它可能会被包含在对 $B^{out}(v,r)$ 的递归调用中。在递归调用中，我们有 $|B^{out}(v,r)| \le |B^{out}(v,R)| \le \frac{7n}{8}$，因为 $r \le R$（相反情况下算法会失败，且不会返回任何边），并且根据引理 53，v 是出轻的。

结合这些情况，我们得到如下关于 $p(n,w,D)$ 的递归。在计算中，我们记 $q := R^{-1}\cdot 10\log(n_0)$：

$$\begin{aligned}
p(n,w,D) &\le \max_{v\in V(G)} \mathbf{P}_{r\sim\mathrm{Geom}(q)}(y \notin B^{out}(v,r) \mid x \in B^{out}(v,r)) + p(\tfrac{7n}{8}, w, D)\\
&\le \max_{v\in V(G)} \mathbf{P}_{r\sim\mathrm{Geom}(q)}(r < \mathrm{dist}(v,y) \mid r \ge \mathrm{dist}(v,x)) + p(\tfrac{7n}{8}, w, D)\\
&\le \max_{v\in V(G)} \mathbf{P}_{r\sim\mathrm{Geom}(q)}(r < \mathrm{dist}(v,x) + w \mid r \ge \mathrm{dist}(v,x)) + p(\tfrac{7n}{8}, w, D)
\end{aligned}$$

令 $r' := r - \mathrm{dist}(v,x)$。在事件 $r \ge \mathrm{dist}(v,x)$ 的条件下，r' 是一个非负随机随机变量，并且根据几何分布的无记忆性，r' 也从几何分布(q) 中采样：

$$\begin{aligned}
&\le \max_{v\in V(G)} \mathbf{P}_{r'\sim\mathrm{Geom}(q)}(r < w) + p(\tfrac{7n}{8}, w, D)\\
&\le wq + p(\tfrac{7n}{8}, w, D)\\
&\le \frac{w}{D}\cdot 100\log(n_0)\log(n) + p(\tfrac{7n}{8}, w, D).
\end{aligned}$$

在最后一步中，我们代入了 $q = R^{-1}\cdot 10\log(n_0) = \frac{1}{D}\cdot 100\log(n_0)\log(n)$。根据归纳法可得，$p(n,w,D) \le \frac{w}{D}\cdot 100\log(n_0)\log(n)\log^{8/7}(n) \le \frac{w}{D}\cdot 1000\log(n_0)\log^2(n)$。

同样的分析也适用于将“B^{out}”替换为“B^{in}”的入球。

重情形。在重情形中，算法选取一个重顶点 v，计算集合 $W = B^{out}(v,R)\cap B^{in}(v,R)$ 和 $C \supseteq W$，并在图 G/C 上递归；该图是将顶点集 C 收缩为单个顶点后得到的。由于 v 是重顶点，根据引理 53，我们有 $|B^{out}(v,R)|, |B^{in}(v,R)| > \frac{3n}{4}$。由此可得 $|C| \ge |W| > \frac{n}{2}$，因此收缩后的

图的规模为 $|V(G/C)| \le \frac{n}{2}$。由于我们以参数 $D-4R$递归调用算法，其中 $R = {}_{10}{}^{D}{}_{\log n}$，因此得到以下递推式：

$$p(n,w,D) \le p(\tfrac{n}{2}, w, D-4R).$$

使用归纳法假设，我们得到：

$$\begin{aligned} & p(n,w,D) \\ & \quad \le \frac{w}{D-4R} \cdot 1000\log^2(n_0)\log(\tfrac{n}{2}) \\ & \quad \le \frac{w}{D} \cdot \frac{1}{1-\frac{4}{10\log n}} \cdot 1000\log^2(n_0)\log(\tfrac{n}{2}) \\ & \quad = \frac{w}{D} \cdot \frac{\log(n)}{\log(n)-\frac{4}{10}} \cdot 1000\log^2(n_0)\cdot(\log(n)-1) \\ & \quad \le \frac{w}{D} \cdot 1000\log^2(n_0)\cdot\log(n). \end{aligned}$$

◀

▶ **引理 56**(算法 9 的运行时间)。 算法 $\textsc{StrongLDD}(G,\ D)$ 运行时间为 $O((m+n_0\log\log n_0)\log^2(n_0))$。

证明。 首先只关注算法的一次调用，忽略递归调用的开销。计算重轻分类需要时间 $O((m+n_0\log\log n_0)\log(n_0))$。在重情形下，我们可以在 Dijkstra 算法时间 $O(m+n_0\log\log n_0)$ 内计算 $W,\ T_{out}T^{in},\ C$。在轻情形下，我们也可以在总时间 $O(m+n_0\log\log(n_0))$ 内划出所有球 $B^{out}(v,\ r)$ 和 $B^{in}(v,\ r)$，尽管形式化分析更加复杂：注意到，我们至多探索每个顶点一次，每次耗时 $O(\log\log n_0)$，并且至多探索每条边一次，每次耗时 $O(1)$。由于分析与引理 8 类似，我们省略进一步细节。

由于算法会在 G的不相交子图上递归，而每个子图的节点数至多是原节点数的某个常数因子，因此运行时间变为 $O((m+n_0\log\log n_0)\log(n_0)^2)$。◀

▶ **引理 57** (算法 9 的失败概率)。 $\textsc{StrongLDD}(G,\ D)$ 返回 失败 的概率至多为 $O(n_0^{-8})$。

证明。正如前面的引理中详细说明的那样，每次递归调用都会使顶点数减少一个常数因子，因此递归深度至多为 $O(\log n_0)$。在每次递归调用中，第 13 行和第 18 行的循环最多运行 n_0 次。每次执行时，误差事件为 $r > R$，其中 $r \sim$ 服从几何分布 $\mathrm{Geom}(R^{-1}\cdot 10\log(n_0))$。该事件发生的概率至多为 $\exp(-10\log(n_0)) \le n_0^{-10}$，因此算法返回失败的概率至多为 $O(n_0\log n_0)\cdot n_0^{-10} \le O(n_0^{-8})$。◀

定理 5 的证明。为了计算所述的强低直径分解，我们调用 $\textsc{StrongLDD}(G, \frac{1}{2}D)$并进行以下两项修改：

首先，每当某个递归调用返回失败时，我们就简单地重新启动整个算法 .

其次，我们测试返回的边集 $S \subseteq E(G)$ 是否满足强直径性质。为此，我们计算 $G\setminus S$ 中的强连通分量，并针对任意这样的分量 C，计算其直径的一个 2-近似值。通过一个标准论证，这样的 2-近似值可以在 Dijkstra 算法时间内获得，具体方法是：(1) 选择任意节点 v；(2) 通过在 G 上求解单源最短路径，计算 $d^{out} := \max_{u\in V(G)} d_G(v,u)$；(3) 通过在 G 的反向图上求解单源最短路径，计算 $d^{in} := \max_{u\in V(G)} d_G(u,v)$，并返回 $\max\{d^{in}, d^{out}\}$。如果所有分量中的直径近似值均至多为 $\frac{D}{2}$，则返回 S。否则，我们重新启动整个算法。

该算法确实永远不会无法满足强直径性质：由于直径近似的近似因子至多为 2，我们已验证图 $G \setminus S$ 中任意强连通分量的直径至多为 D。此外，以高概率，算法 9 的执行会以高概率通过两项测试（根据引理 54 和 57），因此我们预计重复运行该算法 $O(1)$ 次。由于各次重复相互独立，边命中概率仅增加一个常数因子，并根据引理 55 保持为 $O(\frac{w(e)}{D} \cdot \log^3(n_0))$。

最后，考虑运行时间。正如之前所论证的，我们以高概率完全避免重新启动算法 9。因此，根据引理 56，该算法以高概率在总时间 $O((m+n_0 \log\log n_0)\log^2(n_0))$ 内运行。由于我们预计最多重复该算法 $O(1)$ 次，因此相同的界也适用于期望运行时间。◀

一种惰性 Dijkstra 算法

本节旨在证明以下引理，该引理指出，可以调整Dijkstra 算法，使其能够处理负权边，运行时间取决于 $\eta_G(v)$的值。回顾一下，$\eta_G(v)$表示 G 中一条最短 s-v 路径所包含的负权边数量的最小值。

▶ **引理 25**（带负权的Dijkstra算法，类似于 [11, 引理 3。3]）。设 G 为一个带有源顶点 $s \in V$（G）且不包含负环的有向图。存在一种算法，可以在 O（$\sum_v$（deg(v) $+\log\log n$）$\cdot \eta_G$（v））时间内从 s 计算最短路径树。（如果 G 包含负环，该算法不会终止。）

这个引理基本上就是 [11, 引理 3.3], ，但表述略有不同。我们给出一个自成一体的证明，其思路大致遵循 [11, 附录 A] 中的证明。

我们在算法10 中给出引理 25 的伪代码。在全文中，令 $G = (V, E, w)$ 表示给定的、边权可能为负的有向加权图。我们将 $E^{\geq 0}$ 写作具有非负权重的边的子集，将 $E^{<0}$ 写作具有负权重的边的子集。在伪代码中，我们依赖 Thorup 优先队列：

▶**引理 58** (Thorup 优先队列 [57])。有一种用于存储 n 整数键的优先队列实现，支持常数时间内执行 *FindMin*、*Insert* 和 *DecreaseKey* 操作，并在 $O(\log\log n)$时间内执行 *Delete* 操作。

对于该算法的分析，我们定义两个核心量。令 v 为一个顶点，则我们定义

$$\text{dist}_i(v) = \min\{\, w(P) : P\text{是包含少于 } i \text{ 条负权边的 } s\text{-}v\text{-路径} \,\},$$

$$\text{dist}'_i(v) = \min\left\{ \text{dist}_i(v),\ \min_{\substack{u \in V \\ w(u,v)<0}} \text{dist}_i(u) + w(u,v) \right\}.$$

注意，$\text{dist}_0(v) = \text{dist}'_0(v) = \infty$。我们先从涉及这些量$\text{dist}_i$ 和 dist'_i的一些观察开始：

▶ **Observation 59.** *For all i,* $\text{dist}_i(v) \geq \text{dist}'_i(v) \geq \text{dist}_{i+1}(v)$.

▶**观察 60.** 对于所有 v,

$$\text{dist}_{i+1}(v) = \min\left\{ \text{dist}_i(v),\ \min_{\substack{u \in V \\ \text{dist}_i(u) > \text{dist}'_i(u)}} \text{dist}'_i(u) + \text{dist}_{G^{\geq 0}}(u,v) \right\}.$$

算法 10 实现引理 25 的 Dijkstra 算法版本。

为所有顶点 $v \neq s$ 初始化 $d[s] \leftarrow 0$ 和 $d[v] \leftarrow \infty$
初始化一个 Thorup优先队列 Q，其键为 $d[\cdot]$，并将 s 加入 Q
重复
    （*Dijkstra* 阶段）
    $A \leftarrow \emptyset$
    当 Q 非空时**执行**
        从 Q 中移除具有最小 $d[v]$ 的顶点 v
        添加 v 至 A
        对于每条边 $(v,x) \in E^{\geq 0}$，**执行**
            如果 $d[v]+w(v,w) < d[x]$ **那么**
                将 x 添加到 Q
                $d[x] \leftarrow d[v]+w(v,\ x)$
    （贝尔曼-福特阶段）
    对每个 $v \in A$ **执行**
        对每条边 $(v,\ x) \in E^{<0}$ **执行**
            如果 $d[v]+w(v,\ x) < d[x]$，**则**
                将 x 添加到 Q
                $d[x] \leftarrow d[v]+w(v,\ x)$
直到 Q 为空
返回 $d[v]$ 对于所有顶点 v

证明。若 $\mathrm{dist}_i(v) = \mathrm{dist}_{i+1}(v)$，则该命题显然成立，因此假设 $\mathrm{dist}_{i+1}(v) < \mathrm{dist}_i(v)$。令 P 为一条见证 $\mathrm{dist}_{i+1}(v)$ 的路径，即一条包含少于 $i+1$ 条负权边的最短 s-v 路径。令 (x,u) 表示 P 中最后一条负权边，并将路径 P 划分为若干子路径 $P_1\,x\,u\,P_2$。于是，第一段 $P_1\,x$ 是一条包含少于 i 条负权边的路径，而路径段 $u\,P_2$ 不包含任何负权边。因此，

$$\mathrm{dist}_{i+1}(v) = \mathrm{dist}_i(x) + w(x,u) + \mathrm{dist}_{G^{\geq 0}}(u,v) \geq \mathrm{dist}'_i(u) + \mathrm{dist}_{G^{\geq 0}}(u,v).$$

假设，为了导出矛盾，$\mathrm{dist}_i(u) = \mathrm{dist}'_i(u)$。那么

$$\mathrm{dist}_{i+1}(v) \geq \mathrm{dist}_i(u) + \mathrm{dist}_{G^{\geq 0}}(u,v) \geq \mathrm{dist}_i(v),$$

这与我们的初始假设相矛盾。 ◀

▶**观察 61**。对于所有 v，

$$\mathrm{dist}'_i(v) = \min\left\{ \mathrm{dist}_i(v),\ \min_{\substack{u\in V \\ \mathrm{dist}_{i-1}(u) > \mathrm{dist}_i(u) \\ w(u,v)<0}} \mathrm{dist}_i(u) + w(u,v) \right\}$$

证明。若 $\mathrm{dist}_i(v) = \mathrm{dist}'_i(v)$，则结论显然成立，因此假设 $\mathrm{dist}'_i(v) < \mathrm{dist}_i(v)$。于是存在某个顶点 $u \in V$，使得 $w(u,v) < 0$，且 $\mathrm{dist}'_i(v) = \mathrm{dist}_i(u) + w(u,v)$。只需证明 $\mathrm{dist}_{i-1}(u) > \mathrm{dist}_i(u)$。反设 $\mathrm{dist}_{i-1}(u) = \mathrm{dist}_i(u)$。那么 $\mathrm{dist}'_i(v) = \mathrm{dist}_{i-1}(u) + w(u,v) \geq \mathrm{dist}'_{i-1}(v)$，这与我们的初始假设矛盾（根据观察 59）。◀

▶ **引理 62** (算法 10 的不变量)。 考虑算法 *10* 中循环的第 i 次迭代（从1开始）。此时以下不变量成立：
1. Dijkstra 阶段结束后（第 *11* 行之后）：*a.* 对所有顶点 v，$d[v] = \mathit{dist}_i(v)$；*b.* $A = \{ v : \mathit{dist}_{i-1}(v) > \mathit{dist}_i(v) \}$。*2. Bellman-Ford* 阶段结束后（第 *16* 行之后）：*a.* 对所有顶点 v，$d[v] = \mathit{dist}'_i(v)$；*b.* $Q = \{ v : \mathit{dist}_i(v) > \mathit{dist}'_i(v) \}$。

证明。我们对 i使用归纳法证明这些不变量。

第一个 Dijkstra 阶段 我们从分析第一次迭代 $i = 1$ 开始。Dijkstra 阶段的执行过程与常规 Dijkstra 算法完全相同。因此，$d[v] = \mathrm{dist}_{G^{\geq 0}}(s, v) = \mathrm{dist}_1(v)$，这正如不变量 1a 所述。此外，我们在 A 中恰好包含了所有在 $G^{\geq 0}$ 中可从 s 到达的顶点。事实上，对于这些顶点 v，我们有 $\mathrm{dist}_1(v) = \mathrm{dist}_{G^{\geq 0}}(s, v) < \infty$，且 $\mathrm{dist}_0(v) = \infty$，因此 $A = \{ v : \mathrm{dist}_0(v) > \mathrm{dist}_1(v) \}$，这就证明了不变量 1b。

后续的 Dijkstra 阶段 接下来，我们分析后续迭代中的 Dijkstra 阶段，$i > 1$。令 d' 表示数组 d 在 Dijkstra 阶段之后的状态，我们的目标是证明$d'[v] = \mathrm{dist}_i(v)$ 对所有顶点 v均成立。因此，任取一个顶点 v；我们可以假设 $\mathrm{dist}_{i+1}(v) < \mathrm{dist}_i(v)$，否则，利用算法永远不会增大 $d[\cdot]$，即可轻松证明该结论。对 Dijkstra 算法的标准分析表明

$$d'[v] = \min_{u \in Q} (d[u] + \mathrm{dist}_{G^{\geq 0}}(u, v)),$$

其中 Q 是执行 Dijkstra 算法之前的队列。代入归纳假设和观察 60 后，我们确实可以得出

$$d'[v] = \min_{\substack{u \in V \\ \mathrm{dist}_{i-1}(v) > \mathrm{dist}'_{i-1}(v)}} d[u] + \mathrm{dist}_{G^{\geq 0}}(u, v) = \mathrm{dist}_i(v),$$

这就证明了不变量 1a。

为分析不变量 1b 和集合 A，首先回顾一下：在执行 Dijkstra 阶段之前，我们会将 A 重置为空集。之后，我们会向 A 中添加且仅添加以下顶点：(i) 最初包含在队列 Q 中的顶点，或 (ii) 满足 $d'[v] < d[v]$的顶点。注意，根据观察 59，这些集合恰好是 (i) $\{ v : \mathrm{dist}_i(v) > \mathrm{dist}'_i(v) \}$ 和 (ii) $\{ v : \mathrm{dist}'_{i-1}(v) > \mathrm{dist}_i(v) \}$，其并集恰好为 $\{ v : \mathrm{dist}_{i-1}(v) > \mathrm{dist}_i(v) \}$。

Bellman-Ford 阶段 对 Bellman-Ford 阶段的分析更为简单。再次将 d'数组的状态记为 d Bellman-Ford 阶段执行后，根据观察 61，我们有

$$d'[v] = \min_{\substack{u \in A \\ w(u,v) < 0}} d[u] + w(u, v) = \min_{\substack{u \in V \\ \mathrm{dist}'_{i-1}(u) > \mathrm{dist}_i(u) \\ w(u,v) < 0}} \mathrm{dist}_i(u) + w(u, v) = \mathrm{dist}'_i(v),$$

这证明了不变式 2a。这里我们再次假设 $\mathrm{dist}'_i(v) < \mathrm{dist}_i(v)$，否则该陈述显然成立，因为算法从不会增加 $d[\cdot]$。

此外，在Dijkstra 阶段终止后，队列 Q 为空。之后，在当前的Bellman-Ford 阶段中，我们恰好将满足 $\text{dist}_i(v) > \text{dist}'_i(v)$ 的那些顶点 v 插入队列，因此 $Q = \{ v : \text{dist}_i(v) > \text{dist}'_i(v)\}$，这证明了不变式 2b。◀

根据这些不变量（以及前面的观察），我们可以轻松得出算法 10 的正确性：

▶ **引理 63**（算法 10 的正确性）。如果给定的图 G 包含负环，则算法 *10* 不会终止。此外，如果算法 *10* 终止，则它已正确计算出 $d[v] = \text{dist}_G(s, v)$。

证明。我们证明，算法终止后，所有边 (u, v) 都已松弛，即 $d[v] \le d[u] + w(u, v)$。确实，假设存在一条边 (u, v) 尚未松弛，即 $d[v] > d[u] + w(u, v)$。令 i 表示算法的最后一次迭代。根据不变式 2a，我们有 $d[x] = \text{dist}'_i(x)$；根据不变式 2b，我们有 $\text{dist}'_i(x) = \text{dist}_i(x)$（假设 $Q = \emptyset$），对所有顶点 x 均成立。我们分两种情况讨论：如果 $w(u, v) \ge 0$，那么我们有 $\text{dist}_i(v) > \text{dist}_i(u) + w(u, v)$—— 矛盾。若 $w(u, v) < 0$，那么我们有 $\text{dist}'_i(v) = \text{dist}_i(u) + w(u, v)$——同样矛盾。

到目前为止，我们已经证明，如果算法终止，所有边都已松弛。容易验证，如果 G 包含负环，那么该环中至少有一条边无法松弛。因此，只要 G 包含负环，算法就不会终止。

相反，假设 G 不包含负环。我们声称算法已正确计算出所有距离。首先，回顾一下，我们始终有 $d[v] \ge \text{dist}_G(s, v)$。考虑任意一条最短的 s-v-路径 P；我们通过对 P的长度作归纳法，证明 $d[v] = w(P)$。当 $|P| = 0$时，我们最初正确地设置了 $d[s] = 0$。（注意，$\text{dist}_G(s, s)$不可能为负，否则 G 会包含负环。）因此，假设 P 非空，且 P可以写成 $P_1\, u\, v$。根据归纳假设，$d[u] = \text{dist}_G(P_1\, u)$。由于边（$u, v$）已松弛，我们有 $d[v] \le d[u] + w(u, v) = w(P) = \text{dist}_G(s, v)$。回顾一下，我们还有 $d[v] \ge \text{dist}_G(s, v)$，因此 $d[v] = \text{dist}_G(s, v)$。◀

对我们而言，证明中最相关的变化是运行时间分析。回顾一下，$\eta_G(v)$ 表示一条最短 s-v 路径中负权边的最小数量，而 $\deg(v)$ 表示顶点 v 的出度。

▶ **引理 64**（算法 10 的运行时间）。假设 G 不包含负环。那么算法 *10* 的运行时间为 $O(\sum_v (\deg(v) + \log\log n)\eta_G(v))$。

证明。考虑算法的一次迭代。令 A 表示集合 A在该迭代结束时（即 Dijkstra 阶段结束时）的状态，则整个迭代的运行时间可界定为：

$$O\left(\sum_{v \in A} (\deg(v) + \log\log n)\right).$$

确实，在 Dijkstra 阶段的每次迭代中，删除队列中的一个元素需要花费 $O(\log\log n)$ 的时间（引理 58），但对于每次这样的删除，我们都会在 Q 中向 A 添加一个新元素。此外，在 Dijkstra 阶段和 Bellman-Ford 阶段，我们都只枚举从 A 中某个顶点出发的边，总计涉及 $O(\sum_{v \in A} \deg(v))$ 条边。循环的内部步骤（第 9 至 11 行以及第 14 至 16 行）每一步的运行时间均为常数（引理 58）。

令 A_i 表示 i 在第 i 次迭代时的状态。那么总运行时间为

$$O\left(\sum_{i=1}^{\infty}\sum_{v\in A_i}(\deg(v)+\log\log n)\right)=O\left(\sum_{v\in V}|\{\,i:v\in A_i\,\}|\cdot(\deg(v)+\log\log n)\right).$$

为了完成证明，只需证明 $|\{\,i\colon v\in A_i\,\}|\le\eta_G(v)$。为此，我们首先注意到 $\mathrm{dist}_{\eta_G(v)+1}(v)=\mathrm{dist}_{\eta_G(v)+2}=\cdots=\mathrm{dist}_G(s,v)$。由于根据上述不变量，我们知道 $A_i=\{\,v\colon \mathrm{dist}_{i-1}(v)>\mathrm{dist}_i(v)\,\}$，因此 v 只能包含在集合 $A_1,\dots,A_{\eta_G(v)}$ 中。◀

结合引理 63 和引理 64，完成了引理 25 的证明。

参考文献

1 Ravindra K. Ahuja、Thomas L. Magnanti 和 James B. Orlin。网络流——理论、算法与应用。Prentice Hall，1993。1.1.1 2 Ravindra K. Ahuja、Kurt Mehlhorn、James B. Orlin 和 Robert Endre Tarjan。更快的最短路径问题算法。《ACM 杂志》，37(2):213–223，1990。doi:10.1145/77600.77615。1 3 Baruch Awerbuch。网络同步的复杂性。《ACM 杂志》，32(4):804–823，1985。doi:10.1145/4221.4227。1.1.2 4 Baruch Awerbuch、Bonnie Berger、Lenore Cowen 和 David Peleg。快速网络分解（扩展摘要）。载于 Norman C. Hutchinson 编，ACM 第 11 届年度分布式计算原理研讨会（PODC 1992），第 169–177 页。ACM，1992。doi:10.1145/135419.135456。1.1.2 5 Baruch Awerbuch、Andrew V. Goldberg、Michael Luby 和 Serge A. Plotkin。分布式计算中的网络分解与局部性。载于第 30 届年度计算机科学基础研讨会（FOCS 1989），第 364–369 页。IEEE 计算机学会，1989。doi:10.1109/SFCS.1989.63504。1.1.2 6 Baruch Awerbuch 和 David Peleg。具有多项式通信空间权衡的路由。SIAM 离散数学杂志，5(2):151–162，1992。 doi:10.1137/0405013。1.1.2 7 Kyriakos Axiotis、Aleksander Madry 和 Adrian Vladu。用于加速单位容量图上最小费用流的环流控制。载于 Sandy Irani 编，第 61 届年度计算机科学基础研讨会（FOCS 2020），第 93–104 页。IEEE，2020。 doi:10.1109/FOCS46700.2020.00018。1 8 Yair Bartal。度量空间的概率近似及其算法应用。载于第 37 届年度计算机科学基础研讨会（FOCS 1996），第 184–193 页。IEEE 计算机学会，1996。doi:10.1109/SFCS.1996.548477。1.1.2 9 Richard Bellman。关于一个路由问题。《应用数学季刊》，16(1):87–90，1958。doi:10.1090/qam/102435。1，15 10 Aaron Bernstein、Maximilian Probst Gutenberg 和 Christian Wulff-Nilsen。稠密加权有向图上的近最优递减式单源最短路径。载于 Sandy Irani 编，第 61 届年度计算机科学基础研讨会（FOCS 2020），第 1112–1122 页。IEEE，2020。 doi10.1109/FOCS46700.2020.00107。1.1.2 11 Aaron Bernstein、Danupon Nanongkai 和 Christian Wulff-Nilsen。近线性时间的负权单源最短路径。载于第 63 届年度计算机科学基础研讨会（FOCS 2022），第 600–611 页。IEEE，2022。doi:10.1109/FOCS54457.2022.00063。1，1.1.2，1.1.2，1.2.2，1.2.3，3.1，3.2，3.2，25，3.2，26，3.2，4，5，5.1，5.2，7，7.2，25，A 12 Guy E. Blelloch、Anupam Gupta、Ioannis Koutis、Gary L. Miller、Richard Peng 和 Kanat Tangwongsan。近线性工作量的并行 SDD 求解器、低直径分解与低伸展子图。《理论计算机系统》，55(3):521–554，2014。 doi:10.1007/s00224-013-9444-5。1.1.2

13 Karl Bringmann和Tobias Friedrich。几何随机变量和随机图的精确高效生成。载于Fedor V. Fomin、Rusins Freivalds、Marta Z. Kwiatkowska和DavidPeleg（编），第 40 届国际自动机、语言与编程学术讨论会（ICALP 2013），《计算机科学讲义》第7965卷，第 267–278 页。施普林格，2013 年。 doi:10.1007/978-3-642-39206-1_23。3.1 14 Peter Butkovic 和 Raymond A.Cuninghame-Green。用于 $n \times n$ 双值矩阵最大环均值的 $O(n^2)$算法。离散应用数学，35(2)：157–162，1992 年。 doi:10.1016/0166-218X(92)90039-D。1.1.1 15 Krishnendu Chatterjee、Monika Henzinger、Sebastian Krinninger、Veronika Loitzenbauer 和Michael A.Raskin。最小环均值的近似。理论计算机科学，547：104–116，2014 年。 doi:10.1016/j.tcs.2014.06.031。1.1.116Shiri Chechik和Tianyi Zhang。次多项式时间内的动态低拉伸生成树。载于ShuchiChawla（编），第 31 届 ACM-SIAM离散算法研讨会（SODA2020），第463–475 页。SIAM，2020 年。 doi:10.1137/1.9781611975994.28。1.1.2 17Li Chen、Rasmus Kyng、Yang P. Liu、Richard Peng、Maximilian ProbstGutenberg和 Sushant Sachdeva。近线性时间内的最大流和最小费用流。载于第63 届计算机科学基础研讨会（FOCS2022），第 612–623页。IEEE，2022年。 doi:10.1109/FOCS54457.2022.00064。1 18 BorisV. Cherkassky、Loukas Georgiadis、Andrew V. Goldberg、RobertEndre Tarjan 和RenatoFonseca F. Werneck。最短路径可行性算法：实验评估。载于J.Ian Munro和 Dorothea Wagner（编），第 10 届算法工程与实验研讨会（ALENEX2008），第118–132 页。SIAM，2008 年。 doi:10.1137/1.9781611972887.12。1.1 19 Boris V.Cherkassky 和 Andrew V. Goldberg。负环检测算法。数学规划，85(2)：277–311，1999 年。 doi:10.1007/s101070050058。1.120 Boris V. Cherkassky、AndrewV.Goldberg和Craig Silverstein。桶、堆、列表和单调优先队列。SIAM 计算学报，28(4)：1326–1346，1999 年。 doi:10.1137/S0097539796313490。1 21 Michael B. Cohen、Aleksander Madry、Piotr Sankowski和 AdrianVladu。 $\widetilde{O}(m^{10/7} \log W)$ 时间内的负权最短路径和单位容量最小费用流。载于 PhilipN.Klein（编），第 28 届 ACM-SIAM离散算法研讨会（SODA2017），第 752–771 页。SIAM，2017年。 doi:10.1137/1.9781611974782.48。1 22 Wikipedia 贡献者。最短路径更快算法，2023 年。URL： https://en.wikipedia.org/wiki/Shortest_path_faster_algorithm。1.1 23EdsgerW. Dijkstra。关于与图相关的两个问题的注记。数值数学，1：269–271，1959 年。 doi:10.1007/BF01386390。1，14，7.1 24 JittatFakcharoenphol和Satish Rao。平面图、负权边、最短路径与近线性时间。载于第42 届计算机科学基础研讨会（FOCS 2001），第 232–241 页。IEEE 计算机学会，2001 年。 doi:10.1109/SFCS.2001.959897。1 25 Uriel Feige、Prabhakar Raghavan、David Peleg和 Eli Upfal。利用含噪信息进行计算。SIAM 计算学报，23(5)：1001–1018，1994年。 doi:10.1137/S0097539791195877。1.2.326Lester Ford。网络流理论。论文 P-923，加利福尼亚州兰德公司，1956 年。URL：https://www.rand.org/pubs/papers/P923.html。1，15 27 SebastianForster和Gramoz Goranci。通过动态低直径分解实现动态低拉伸树。载于Moses Charikar和EdithCohen（编），第 51 届 ACM计算理论研讨会（STOC 2019），第 377–388页。ACM，2019 年。 doi:10.1145/3313276.3316381。1.1.2 28 Sebastian Forster、MartinGrösbacher和 Tijn de Vos。用于伸展子图和低直径分解的改进随机移位算法。载于Quentin Bramas、Vincent Gramoli和 Alessia Milani（编），第 25 届分布式系统原理国际会议

(OPODIS 2021), LIPIcs 第 217 卷，页码 16:1–16:17。Schloss Dagstuhl - Leibniz信息学中心，2021。 doi:10.4230/LIPIcs.OPODIS.2021.16。1.1.2 29 Michael L. Fredman 和 Robert Endre Tarjan。斐波那契堆及其在改进网络优化算法中的应用。收录于第 25 届年度计算机科学基础研讨会（FOCS 1984），页码 338–346。IEEE 计算机学会，1984。 doi:10.1109/SFCS.1984.715934。1 30 Michael L. Fredman 和 Dan E. Willard。利用融合树突破信息论界限。《计算机系统与科学杂志》，47(3):424–436，1993。 doi:10.1016/0022-0000(93)90040-4。1 31 Michael L. Fredman 和 Dan E. Willard。用于最小生成树和最短路径问题的跨二分算法。《计算机系统与科学杂志》，48(3):533–551，1994。 doi:10.1016/S0022-0000(05)80064-9。1 32 Harold N. Gabow。网络问题的缩放算法。收录于第 24 届年度计算机科学基础研讨会（FOCS 1983），页码 248–257。IEEE 计算机学会，1983。doi:10.1109/SFCS.1983.68。1, 1.1.1, 1.2.3, 4, 4 33 Harold N. Gabow 和 Robert Endre Tarjan。更快速的网络问题缩放算法。《SIAM计算杂志》，18(5):1013–1036，1989。 doi:10.1137/0218069。1, 1.1.1, 1.2.3, 4, 4 34 Andrew V. Goldberg。最短路径问题的缩放算法。《SIAM计算杂志》，24(3):494–504，1995。 doi:10.1137/S0097539792231179。1, 1.1.1, 1.2.3, 2, 4, 4 35 Donald B. Johnson。稀疏网络最短路径的高效算法。《ACM 杂志》，24(1):1–13，1977。 doi:10.1145/321992.321993。16, 17 36 Richard M. Karp。有向图中最小环均值的刻画。《离散数学》，23(3):309–311，1978。 doi:10.1016/0012-365X(78)90011-0。1.1.1 37 Philip N. Klein、Shay Mozes 和 Oren Weimann。具有负长度的有向平面图最短路径：一种线性空间、 $O(n\log^2 n)$ 时间算法。收录于 Claire Mathieu（编），第 20 届年度 ACM-SIAM离散算法研讨会（SODA 2009），页码 236–245。SIAM，2009。URL: http://dl.acm.org/citation.cfm?id=1496770.1496797。1 38 Philip N. Klein、Satish Rao、Monika Rauch Henzinger 和 Sairam Subramanian。更快速的平面图最短路径算法。收录于 Frank Thomson Leighton 和 Michael T. Goodrich（编），第 26 届年度 ACM计算理论研讨会（STOC 1994），页码 27–37。ACM，1994。 doi:10.1145/195058.195092。1 39 E.L. Lawler。《组合优化：网络与拟阵》。Holt, Rinehart and Winston，1976。1.1.1 40 Eugene L Lawler。双重加权有向线性图中的最优环。收录于国际图论研讨会论文集，页码 209–232，1966。1.1.1 41 Johannes Lengler。漂移分析。CoRR，abs/1712.00964，2017。URL： http://arxiv.org/abs/1712.00964, arXiv:1712.00964。1.2.3, 44 42 Nathan Linial 和 Michael E. Saks。低直径图分解。《组合学》，13(4):441–454，1993。 doi:10.1007/BF01303516。1.1.2 43 Richard J. Lipton、Donald J. Rose 和 Robert Endre Tarjan。广义嵌套剖分。《SIAM数值分析杂志》，16(2):346–358，1979。 doi:10.1137/0716027。1 44 Kurt Mehlhorn 和 Stefan Näher。在 $O(\log\log N)$ 时间和 $O(n)$ 空间内实现有界有序字典。《信息处理快报》，35(4):183–189，1990。 doi:10.1016/0020-0190(90)90022-P。1 45 Gary L. Miller、Richard Peng 和 Shen Chen Xu。利用随机位移进行并行图分解。收录于 Guy E. Blelloch 和 Berthold Vöcking（编），第 25 届 ACM算法与体系结构并行性研讨会（SPAA 2013），页码 196–203。ACM，2013。 doi:10.1145/2486159.2486180。1.1.2 46 Edward F. Moore。穿越迷宫的最短路径。收录于国际开关理论研讨会论文集，页码 285–292。剑桥，哈佛大学出版社，1959。1, 1.1, 15

47 James B. Orlin 和 Ravindra K. Ahuja。用于指派问题和最小平均环问题的新型缩放算法。《数学规划》，54:41–56，1992。 doi:10.1007/BF01586040。1.1.1 48 Jakub Pachocki、Liam Roditty、Aaron Sidford、Roei Tov 和 Virginia Vassilevska Williams。有向图中的环近似：围长与往返距离扩展器的快速算法。收录于 Artur Czumaj（编），第 29 届年度 ACM-SIAM离散算法研讨会（SODA 2018），页码 1374–1392。SIAM，2018。 doi:10.1137/1.9781611975031.91。1.1.2 49 Andrzej Pelc。具有已知错误概率的搜索。《理论计算机科学》，63(2):185–202，1989。 doi:10.1016/0304-3975(89)90077-7。1.2.3 50 Andrzej Pelc。带错误的搜索博弈——与说谎者周旋五十年。《理论计算机科学》，270(1-2):71–109，2002。 doi:10.1016/S0304-3975(01)00303-6。1.2.3 51 Rajeev Raman。优先队列：小型、单调且跨二分。收录于 Josep Díaz 和 Maria J. Serna（编），第 4 届年度欧洲算法研讨会（ESA 1996），《计算机科学讲义》第 1136 卷，页码 121–137。施普林格，1996。 doi:10.1007/3-540-61680-2_51。1 52 Rajeev Raman。单源最短路径问题的近期成果。SIGACT News，28(2):81–87，1997。 doi:10.1145/261342.261352。1 53 Piotr Sankowski。矩阵乘法时间内的最短路径。收录于 Gerth Stølting Brodal 和 Stefano Leonardi（编），第 13 届年度欧洲算法研讨会（ESA 2005），《计算机科学讲义》第 3669 卷，页码 770–778。施普林格，2005。 doi:10.1007/11561071_68。1 54 Alfonso Shimbel。通信网络的结构。收录于信息网络研讨会，页码 199–203。布鲁克林理工学院理工出版社，1955。1, 15 55 Robert Endre Tarjan。深度优先搜索与线性图算法。《SIAM计算杂志》，1(2):146–160，1972。 doi:10.1137/0201010。12 56 Mikkel Thorup。关于RAM优先队列。《SIAM计算杂志》，30(1):86–109，2000。 doi:10.1137/S0097539795288246。1 57 Mikkel Thorup。支持常数时间键值减小操作的整数优先队列与单源最短路径问题。收录于 Lawrence L. Larmore 和 Michel X. Goemans（编），第 35 届年度 ACM计算理论研讨会（STOC 2003），页码 149–158。ACM，2003。doi:10.1145/780542.780566。1, 3, 1.2.2, 1.3, 14, 3.1, 7.1, 58 58 Jan van den Brand、Yin Tat Lee、Yang P. Liu、Thatchaphol Saranurak、Aaron Sidford、Zhao Song 和 Di Wang。稠密实例中近线性时间求解最小费用流、马尔可夫决策过程和 ℓ_1-回归。收录于 Samir Khuller 和 Virginia Vassilevska Williams（编），第 53 届年度 ACM计算理论研讨会（STOC 2021），页码 859–869。ACM，2021。doi:10.1145/3406325.3451108。1 59 Jan van den Brand、Yin Tat Lee、Danupon Nanongkai、Richard Peng、Thatchaphol Saranurak、Aaron Sidford、Zhao Song 和 Di Wang。在中等稠密图上以近线性时间实现二分图匹配。收录于 Sandy Irani（编），第 61 届年度计算机科学基础研讨会（FOCS 2020），页码 919–930。IEEE，2020。 doi:10.1109/FOCS46700.2020.00090。1 60 Peter van Emde Boas。以低于对数的时间和线性空间在森林中保持顺序。《信息处理快报》，6(3):80–82，1977。 doi:10.1016/0020-0190(77)90031-X。1 61 Peter van Emde Boas、R. Kaas 和 E. Zijlstra。高效优先队列的设计与实现。《数学系统理论》，10:99–127，1977。 doi:10.1007/BF01683268。1 62 John W. J. Williams。算法 232——堆排序。《ACM通讯》，7(6):347–349，1964。 doi:10.1145/512274.512284。1